**Contents**

|  |  |  |
| --- | --- | --- |
| [1](#_bookmark0) | [Introduction](#_bookmark0) | 6 |
|  | [1.1 Objectives](#_bookmark1) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 6 |
|  | [1.2 Scope](#_bookmark2) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 7 |
|  | [1.3 Document Structure](#_bookmark3) . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 7 |
| [2](#_bookmark4) | [Definition of Terms and Acronyms](#_bookmark4) | 8 |
|  | [2.1 Acronyms and Abbreviations](#_bookmark5) . . . . . . . . . . . . . . . . . . . . . . . . | 8 |
|  | [2.2 Definition of Terms](#_bookmark6) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 8 |
| [3](#_bookmark7) | [Related Documentation](#_bookmark7) | 9 |
| [4](#_bookmark8) | [Overview and Goals](#_bookmark8) | 10 |
|  | [4.1 Requirements Overview](#_bookmark9) . . . . . . . . . . . . . . . . . . . . . . . . . . | 10 |
|  | [4.2 Quality Goals](#_bookmark10) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 12 |
|  | [4.3 Stakeholders](#_bookmark11) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 12 |
| [5](#_bookmark12) | [Architecture Constraints](#_bookmark12) | 13 |
|  | [5.1 Internal Interfaces](#_bookmark13) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 13 |
|  | [5.2 Distributed Work](#_bookmark14) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 14 |
| [6](#_bookmark15) | [Quality Requirements](#_bookmark15) | 15 |
|  | [6.1 Quality Attributes](#_bookmark16) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 15 |
|  | [6.1.1 AUTOSAR Adaptive Platform Standard](#_bookmark17) . . . . . . . . . . . . | 15 |
|  | [6.1.2 AUTOSAR Adaptive Platform Stack](#_bookmark18) . . . . . . . . . . . . . . | 17 |
|  | [6.1.3 AUTOSAR Adaptive Application](#_bookmark19) . . . . . . . . . . . . . . . . | 18 |
|  | [6.2 Quality Scenarios](#_bookmark20) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 19 |
| [7](#_bookmark21) | [System Scope and Context](#_bookmark21) | 20 |
|  | [7.1 Adaptive Application](#_bookmark23) . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 20 |
|  | [7.2 Dependencies](#_bookmark25) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 20 |
|  | [7.2.1 Crypto Provider](#_bookmark26) . . . . . . . . . . . . . . . . . . . . . . . . . | 20 |
|  | [7.2.2 Operating System](#_bookmark27) . . . . . . . . . . . . . . . . . . . . . . . . | 21 |
|  | [7.2.3 Watchdog](#_bookmark29) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 21 |
|  | [7.3 External Systems](#_bookmark31) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 21 |
|  | [7.3.1 AUTOSAR Adaptive Application](#_bookmark32) . . . . . . . . . . . . . . . . | 21 |
|  | [7.3.2 AUTOSAR Classic Platform](#_bookmark33) . . . . . . . . . . . . . . . . . . . | 22 |
|  | [7.3.3 Third-party Platform](#_bookmark35) . . . . . . . . . . . . . . . . . . . . . . . | 22 |
|  | [7.3.4 Diagnostic Client](#_bookmark36) . . . . . . . . . . . . . . . . . . . . . . . . . | 22 |
|  | [7.3.5 Backend](#_bookmark38) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 22 |
| [8](#_bookmark40) | [Solution Strategy](#_bookmark40) | 23 |
|  | [8.1 Architectural Approach](#_bookmark41) . . . . . . . . . . . . . . . . . . . . . . . . . . . | 23 |
|  | [8.2 Decomposition Strategy](#_bookmark42) . . . . . . . . . . . . . . . . . . . . . . . . . . | 23 |
|  | [8.3 UML Profile](#_bookmark44) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 24 |
|  | [8.4 Technology](#_bookmark46) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 25 |

* + 1. [Implementation Language](#_bookmark47) 25
    2. [Parallel Processing](#_bookmark48) 25
  1. [Design Principles](#_bookmark49) 26
     1. [Leveraging existing standards](#_bookmark50) 26
     2. [SOLID principles](#_bookmark51) 26
     3. [Acyclic Dependencies Principle](#_bookmark52) 27
  2. [Deployment](#_bookmark53) 27
  3. [Verification and Validation](#_bookmark54) 28

1. [Building Block View](#_bookmark55) 29
   1. [Overview](#_bookmark56) 29
      1. [Description pattern](#_bookmark58) 29
   2. [Runtime](#_bookmark59) 30
      1. [Execution Management](#_bookmark60) 30
      2. [State Management](#_bookmark75) 36
      3. [Log and Trace](#_bookmark89) 42
      4. [Core](#_bookmark95) 46
      5. [Operating System Interface](#_bookmark100) 47
   3. [Communication](#_bookmark103) 49
      1. [Communication Management](#_bookmark104) 49
      2. [Network Management](#_bookmark111) 54
      3. [Time Synchronization](#_bookmark114) 56
   4. [Storage](#_bookmark122) 62
      1. [Persistency](#_bookmark123) 63
   5. [Security](#_bookmark148) 70
      1. [Cryptography](#_bookmark149) 70
      2. [Identity and Access Management](#_bookmark174) 94
      3. [Adaptive Intrusion Detection System Manager](#_bookmark177) 96
      4. [Firewall](#_bookmark181) 99
   6. [Safety](#_bookmark184) 101
      1. [Platform Health Management](#_bookmark185) 101
   7. [Configuration](#_bookmark198) 106
      1. [Update and Configuration Management](#_bookmark199) 106
      2. [Registry](#_bookmark209) 114
   8. [Diagnostics](#_bookmark212) 116
      1. [Diagnostic Management](#_bookmark213) 117
2. [Use-Case View](#_bookmark238) 134
   1. [Runtime](#_bookmark240) 135
      1. [Execution Management](#_bookmark241) 135
      2. [State Management](#_bookmark245) 137
   2. [Storage](#_bookmark247) 139
      1. [Persistency](#_bookmark248) 139
   3. [Security](#_bookmark255) 145
      1. [Firewall](#_bookmark256) 145
3. [Runtime View](#_bookmark257) 147

|  |  |  |
| --- | --- | --- |
| [11.1 Runtime](#_bookmark258) | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 147 |
| [11.1.1](#_bookmark259) | [Execution Management](#_bookmark259) . . . . . . . . . . . . . . . . . . . . . | 147 |
| [11.1.2](#_bookmark266) | [State Management](#_bookmark266) . . . . . . . . . . . . . . . . . . . . . . . | 152 |
| [11.2 Storage](#_bookmark273) | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 154 |
| [11.2.1](#_bookmark274) | [Persistency](#_bookmark274) . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 154 |
| [11.3 Security](#_bookmark287) | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 158 |
| [11.3.1](#_bookmark288) | [Firewall](#_bookmark288) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 158 |

1. [Deployment View](#_bookmark293) 160
   1. [Vehicle Software Deployment](#_bookmark294) 160
   2. [Deployment of Software Packages on a Machine](#_bookmark295) 161
2. [Cross-cutting Concepts](#_bookmark297) 163
   1. [Overview of Platform Entities](#_bookmark298) 163
   2. [Function Group](#_bookmark300) 164
   3. [Function Group State](#_bookmark301) 164
   4. [Software Cluster](#_bookmark302) 164
   5. [Machine](#_bookmark305) 167
   6. [Manifest](#_bookmark306) 167
   7. [Application Design](#_bookmark307) 168
   8. [Execution Manifest](#_bookmark308) 169
   9. [Service Instance Manifest](#_bookmark309) 169
   10. [Machine Manifest](#_bookmark310) 170
   11. [Diagnostics deployment](#_bookmark311) 170
   12. [Error Handling](#_bookmark314) 172
   13. [Trusted Platform](#_bookmark315) 172
   14. [Secure Communication](#_bookmark316) 173
3. [Risks and Technical Debt](#_bookmark317) 174
   1. [Risks](#_bookmark318) 174
      1. [Risk Assessment](#_bookmark319) 174
      2. [Risk List](#_bookmark321) 175
   2. [Technical Debt](#_bookmark322) 175
4. **Introduction**

This explanatory document provides detailed technical description of the software ar- chitecture of the AUTOSAR Adaptive Platform standard with the main focus on the architecture model.

* 1. **Objectives**

This document is an architecture description of the AUTOSAR Adaptive Platform in accordance to [[1](#_bookmark323), ISO/IEC 42010] and has the following main objectives:

* Identify the **stakeholders** of the AUTOSAR Adaptive Platform and their **con- cerns**.
* Identify the **system scope** and provide **overview information** of the AUTOSAR Adaptive Platform.
* Provide definitions for all used **architecture viewpoints** and a **mapping of all stakeholder concerns to those viewpoints**.
* Provide an **architecture view** and its **architecture models** for each architecture viewpoint used in this architecture description.
* Provide **correspondence rules** and **correspondences** among the contents of this architecture description.
* Provide an **architecture rationale** (explanation, justification, reasoning for de- cisions made) on a high level. A more in-depth documentation of decisions is provided in [[2](#_bookmark324), EXP\_SWArchitecturalDecisions].
* Provide a **record of known inconsistencies and gaps** among the architecture description.

There is some potential for ambiguity about the term "architecture". Association with this term is quite different e.g., for a mass production project in contrast with Adap- tive Platform standardization. For system development of an automotive embedded computer the software architecture usually defines the details of the structural and the behavioral architecture views down to module level. In contrast the architecture of AU- TOSAR Adaptive Platform lacks such details deliberately to provide more degrees of freedom for stack vendors in their solution design.

Beyond the specification of APIs the term "architecture" for Adaptive Platform refers to guidelines how to apply the standard to concrete development projects.

This document describes the original architectural design of the AUTOSAR Adaptive Platform including details how the building blocks should interact with each other. It is an example how an implementation of the standard should work internally. However, a stack vendor is free to choose another design as long as it complies with the binding AUTOSAR Adaptive Platform standard.

* 1. **Scope**

This explanatory document applies to the AUTOSAR Adaptive Platform. It is recom- mended to get an overview of the AUTOSAR Adaptive Platform for all members of the working groups, stack vendors, and application developers.

* 1. **Document Structure**

This document is organized as follows. Section [4](#_bookmark8) provides an overview of the main requirements for the AUTOSAR Adaptive Platform, the top quality goals of its architec- ture, and a list of stakeholders that are affected by it. Section [5](#_bookmark12) lists requirements that constrain design and implementation decisions or decisions about the development process.

Section [6](#_bookmark15) is the base for discovering trade-offs and sensitivity points in the architecture of the AUTOSAR Adaptive Platform by introducing a quality attribute tree followed by the most important quality scenarios. The system context in which the AUTOSAR Adaptive Platform is intended to be used is outlined in section [7](#_bookmark21). Section [8](#_bookmark40) summarizes the fundamental decisions and solution strategies, that shape the architecture of the AUTOSAR Adaptive Platform such as technology decisions or architectural patterns to be used.

Sections [9](#_bookmark55) to [12](#_bookmark293) explain the software architecture from different view points. First, sec- tion [9](#_bookmark55) explains the decomposition of the AUTOSAR Adaptive Platform into Functional Clusters and their interdependencies. Then, section [11](#_bookmark257) demonstrates how the main use cases are realized using the Functional Clusters in the AUTOSAR Adaptive Plat- form. Section [12](#_bookmark293) shows different scenarios how applications based on the AUTOSAR Adaptive Platform may be deployed.

Section [13](#_bookmark297) provides an overview of concepts and patterns used by the AUTOSAR Adaptive Platform. Section [14](#_bookmark317) lists and rates risks associated with the architecture of the AUTOSAR Adaptive Platform and technical debt.

1. **Definition of Terms and Acronyms**
   1. **Acronyms and Abbreviations**

|  |  |
| --- | --- |
| **Abbreviation / Acronym** | **Description** |
| DoIP | Diagnostics over Internet Protocol |
| POSIX | Portable Operating System Interface |
| SecOC | AUTOSAR Secure Onboard Communication |
| TLS | Transport Layer Security |
| UML | Unified Modeling Language |

* 1. **Definition of Terms**

This section lists terms that are specific to this document. A list of general terms for AUTOSAR is provided in the [[3](#_bookmark325), glossary].

|  |  |
| --- | --- |
| **Term** | **Description** |
| Functional Cluster | A logical group of functionality within the AUTOSAR Adaptive  Platform. Functional Clusters are the second level of ab- straction in the building block view (cf. Chapter [9](#_bookmark55)). They are also subject of the individual specification documents that make up the AUTOSAR Adaptive Platform standard. |
| Function Group | A set of modeled Processes. See Section [13.2](#_bookmark300) for details. |
| Thread | The smallest sequence of instructions the can be managed in-  dependently by a scheduler. Multiple Threads can be exe- cuted concurrently within one Process sharing resources such as memory. |
| Watchdog | An external component that supervises execution of the AU-  TOSAR Adaptive Platform. See Section [7.2.3](#_bookmark29) for details. |

1. **Related Documentation**

This document provides a high-level overview of the AUTOSAR Adaptive Platform ar- chitecture. It is closely related to general requirements for AUTOSAR Adaptive Plat- form specified in [[4](#_bookmark326), RS\_Main] and [[5](#_bookmark327), RS\_General], and the architectural decisions documented in [[2](#_bookmark324), EXP\_SWArchitecturalDecisions].

The individual building blocks of the architecture (Functional Clusters) are spec- ified in separate documents. Each Functional Cluster defines one or more requirements specification(s) (RS document), one or more software specification(s) (SWS document) and one or more explanatory document(s) (EXP document). Please refer to these documents for any details on the AUTOSAR Adaptive Platform standard.

1. **Overview and Goals**

In conventional automotive systems ECUs are used to replace or augment electro- mechanical systems. Those resource constrained, deeply-embedded ECUs typically perform basic control functions by creating electrical output signals (e.g. for actors) based on input signals (e.g. from sensors) and information from other ECUs con- nected to the vehicle network. Much of the control software is specifically designed and implemented for the target vehicle and does not change significantly during vehi- cle lifetime. The AUTOSAR Classic Platform standard addresses the needs of these deeply-embedded systems.

Recent and future vehicle functions, such as highly automated driving, will introduce complex and computing resource demanding software that shall fulfill strict safety, in- tegrity and security requirements. Such software performs for example, environment perception and behavior planning, and interacts with external backend and infrastruc- ture systems. The software in the vehicle regularly needs to be updated during the life-cycle of the vehicle, due to evolving external systems, improved or added function- ality, or security problems. The AUTOSAR Classic Platform standard cannot fulfill the needs of such systems. Therefore, AUTOSAR specifies a second software platform, the AUTOSAR Adaptive Platform. It provides high-performance computing and com- munication mechanisms as well as a flexible software configuration, for example, to support software update over-the-air. Features that are specifically defined for the AU- TOSAR Classic Platform, such as access to electrical signals and automotive specific bus systems, can be integrated into the AUTOSAR Adaptive Platform but is not in the focus of standardization.

* 1. **Requirements Overview**

This section provides an overview of the basic requirements for the AUTOSAR Adap- tive Platform that impact its architecture. The corresponding requirement identifiers are provided in square brackets. Please refer to [[4](#_bookmark326), RS\_Main] and [[5](#_bookmark327), RS\_General] for any details, rationale or intended use-cases of these requirements.

**Support of state-of-the-art Technology**

The AUTOSAR Adaptive Platform aims to support resource-intensive (memory, cpu) applications on state-of-the-art hardware. Therefore, the AUTOSAR Adaptive Platform shall support high performance computing platforms [RS\_Main\_00002] as well as vir- tualized environments [RS\_Main\_00511]. The AUTOSAR Adaptive Platform shall be able to run multiple applications in parallel [RS\_Main\_00049], each with concurrent application internal control flows [RS\_Main\_00050].

**Software Update and Configuration**

The AUTOSAR Adaptive Platform shall support a flexible (configuration) data and soft- ware update. Hereby, AUTOSAR Adaptive Platform shall support up- and download of such update packages [RS\_Main\_00650] and change of communication and applica- tion software at runtime [RS\_Main\_00503].

AUTOSAR shall provide a unified way to describe software systems deployed to Adaptive and / or Classic platforms [RS\_Main\_00161]. That kind of description shall also support the deployment and reallocation of AUTOSAR Application Software [RS\_Main\_00150], and shall provide means to describe interfaces of the entire system [RS\_Main\_00160].

**Security**

The AUTOSAR Adaptive Platform shall support the development of secure systems [RS\_Main\_00514] with secure access to ECU data and services [RS\_Main\_00170], and secure onboard communication [RS\_Main\_00510].

**Safety**

The AUTOSAR Adaptive Platform shall support the development of safety related systems [RS\_Main\_00010] that are reliable [RS\_Main\_00011] and highly available [RS\_Main\_00012]. The AUTOSAR Adaptive Platform specifications shall be analyz- able and support methods to demonstrate the achievement of safety related properties accordingly [RS\_Main\_00350].

**Reuse and Interoperability**

The AUTOSAR Adaptive Platform shall support standardized interoperability with non-AUTOSAR software [RS\_Main\_00190] as well as (source code) portability for AUTOSAR Adaptive Applications across different implementations of the platform [RS\_AP\_00111]. Hereby, the AUTOSAR Adaptive Platform shall provide means to describe a component model for application software [RS\_Main\_00080], and support bindings for different programming languages [RS\_Main\_00513].

**Communication**

The AUTOSAR Adaptive Platform shall support standardized automotive communica- tion protocols [RS\_Main\_00280] for intra ECU communication [RS\_Main\_01001] with different network topologies [RS\_Main\_00230].

**Diagnostics**

The AUTOSAR Adaptive Platform shall provide diagnostics means during runtime for production and services purposes [RS\_Main\_00260].

* 1. **Quality Goals**

This section will list the top quality goals for the architecture whose fulfillment is of highest importance to the major stakeholders in a future version of this document. Please refer to the currently un-prioritized list of Quality Attributes in Section [6.1](#_bookmark16).

* 1. **Stakeholders**

This section lists the stakeholders of the AUTOSAR Adaptive Platform architecture and their main expectations.

|  |  |
| --- | --- |
| **Role** | **Expectation** |
| Project Leader | Overview of technical risks and technical debt in the AUTOSAR  Adaptive Platform. |
| Working Group Architecture | Concise yet thorough documentation of the goals and driving  forces of the AUTOSAR Adaptive Platform. Documentation of the original architectural design of the AUTOSAR Adaptive Plat- form standard. Documentation of identified technical risks and technical debt in the AUTOSAR Adaptive Platform. |
| Working Group | Consolidated overview of the AUTOSAR Adaptive Platform ar-  chitecture. Realization of use-cases that span multiple Func- tional Clusters. Usage of interfaces within the AUTOSAR Adaptive Platform. Guidelines and patterns for Functional Cluster and interface design. |
| Stack Developer | Consolidated overview of the original architectural design of the  AUTOSAR Adaptive Platform. Realization of use-cases that span multiple Functional Clusters. Usage of interfaces within the AUTOSAR Adaptive Platform. |
| Application Developer | Overview of the building blocks of the AUTOSAR Adaptive Plat-  form and their purpose and provided functionality. Explanation of the concepts used in the AUTOSAR Adaptive Platform. |
| Integrator | Overview and explanation of the original architectural design of  the AUTOSAR Adaptive Platform. Overview of extension points of the AUTOSAR Adaptive Platform. |

**Table 4.1: Stakeholder table with roles and expectations**

1. **Architecture Constraints**

AUTOSAR is a worldwide development partnership of vehicle manufacturers, suppli- ers, service providers and companies from the automotive electronics, semiconductor and software industry. AUTOSAR standardizes the AUTOSAR Adaptive Platform au- tomotive middleware. The AUTOSAR Adaptive Platform is not a concrete implemen- tation. The AUTOSAR Adaptive Platform standard leaves a certain degree of freedom to implementers of the standard, as most standards do. On the one hand, more free- dom enables competition among the different implementations and a broader choice of properties for users of the AUTOSAR Adaptive Platform. On the other hand, a more strict standardization makes the different implementations compatible and exchange- able (within the standardized scope). Naturally, those attributes are in conflict. It is usually a choice of the standardization organization to evaluate the attributes and de- fine the desired level of strictness.

The AUTOSAR Classic Platform is rather strict in that sense by specifying a detailed layered software architecture imposing many constraints on its implementations. The AUTOSAR Adaptive Platform launched with a less strict approach. That less strict approach puts constraints on the AUTOSAR Adaptive Platform architecture as outlined below.

* 1. **Internal Interfaces**

An important architectural constraint is that only interfaces that are intended to be used by applications or interfaces that are used to extend the functionality of the AUTOSAR Adaptive Platform shall be standardized. Internal interfaces between the building blocks of the AUTOSAR Adaptive Platform shall not be standardized. This approach leaves a lot of freedom to design and optimize the internals of an AUTOSAR Adaptive Platform stack. However, it also imposes constraints on how the AUTOSAR Adaptive Platform architecture can be defined and described in this document. Also, this means that it might not be possible to use different functional clusters from different AUTOSAR Adaptive Platform stack vendors.

First, the existence of internal interfaces and their usage by other building blocks is in most cases a recommendation and reflects the original design approach of the authors of the standard. The same applies to any interactions described in this document that make use of such internal interfaces.

Second, some quality attributes may be hard to ensure in general by the architec- ture of the standard. Additional measures like security or safety considerations lack well-defined inputs such as data flows or specifications of interdependencies. Con- sequently, a more thorough design phase is required when an AUTOSAR Adaptive Platform stack is implemented.

* 1. **Distributed Work**

Standardization of the AUTOSAR Adaptive Platform is a worldwide distributed effort. The individual building blocks are specified by dedicated working groups in separate documents to be able to scale in that distributed setup. This impacts the way the AUTOSAR Adaptive Platform architecture is described in this document.

First, this document shows interfaces on an architectural level only. This document does not specify details of interfaces such as individual operations. This keeps redun- dancies and thus dependencies between this document and the documents actually specifying the individual building blocks manageable. Another consequence is that the interactions shown in this document are not based on actual operations specified in the interfaces but rather on an architectural level as well.

Second, this document aims to provide guidance for the working groups in specify- ing the individual building blocks by defining patterns and concepts to solve common problems. This guidance should help to build a uniform and consistent standard from ground up.

1. **Quality Requirements**

Quality requirements define the expectations of AUTOSAR Adaptive Platform stake- holders for the quality and the attributes of the AUTOSAR Adaptive Platform standard that indicate whether the quality factors are satisfied or not. Section [6.1](#_bookmark16) starts by list- ing the quality attributes that, in the end, are used to assess whether the AUTOSAR Adaptive Platform and its software architecture satisfies the expected quality factors or not. Section [6.2](#_bookmark20) then provides quality scenarios that operationalize quality attributes and turn them into measurable quantities by describing the reaction of the system to a stimulus in a certain situation.

* 1. **Quality Attributes**

The AUTOSAR Adaptive Platform has many stakeholders with different concerns. Thus, this document uses the following three quality attribute categories that corre- spond to the three main stakeholder groups in order to make the requirements and their impact on the architecture more comprehensible:

* **AUTOSAR Adaptive Platform Standard:** Quality requirements for the AU- TOSAR standard itself. These requirements may directly affect the architecture of the AUTOSAR Adaptive Platform.
* **AUTOSAR Adaptive Platform Stack:** Quality requirements for an implementa- tion of the AUTOSAR standard as an AUTOSAR stack. These requirements may indirectly affect the architecture of the AUTOSAR Adaptive Platform.
* **AUTOSAR Adaptive Application:** Quality requirements for an application based on an AUTOSAR stack. These requirements may transitively affect the architec- ture of the AUTOSAR Adaptive Platform.

The quality attributes are organized according to the Architecture Tradeoff Analysis Method (ATAM) [[6](#_bookmark328)] as a tree, one for each of the quality attribute categories. The leafs of those trees are the individual quality attributes.

* + 1. **AUTOSAR Adaptive Platform Standard**
* Functional suitability
  + The software architecture shall reflect the project objectives (POs) and be the consistent source for all specifications (here: completeness with respect to the PO; see also usability below).
  + The standard shall not contain elements that are not traceable to POs, change requests (CRs), or concepts.
  + The standard shall contain at least one element derived from each PO, CR, or concept.
* Performance efficiency
  + The specification shall allow for a run-time efficient implementation. Run- time efficiency refers to all resource consumption, CPU, RAM, ROM.
* Compatibility
  + The standard shall retain older versions of its elements in the face of change.
  + The standard shall be interoperable with pre-existing standards, especially the AUTOSAR Classic Platform. Pre-existing standards means network pro- tocols and the like.
  + The standard shall adopt new versions of pre-existing standards only after an impact analysis.
* Usability
  + The use of the standard shall be as easy as possible for suppliers and appli- cation developers. Easy means: not much material and resources required.
  + The holistic approach shall not be broken (avoid different approaches in one standard).
  + The standard shall contain application sample code for all its elements.
  + The standard shall contain documentation of the use cases for its elements.
  + The standard shall document the semantics of its elements.
  + The standard shall document its decisions, consequences, and implemen- tation restrictions (both for stack & apps) including their rationale.
  + The standards elements shall be easy to use and hard to misuse.
  + The standard shall stick to pre-existing standards, as far as no functional requirements are compromised.
  + The standard shall be as stable as possible.
  + AUTOSAR standards shall not change disruptive but rather evolve evolution- ary (for example, backward-compatibility can be a help).
  + The software architecture shall reflect the PO and be the consistent source for all specifications (here: consistency; see also functional suitability above).
* Reliability
  + The standard shall classify its elements with respect to safety relevance (that is, functional clusters shall be marked if they participate in safety critical operations of the platform).
  + The standard shall specify control flow restrictions between its elements in order to achieve freedom from interference.
  + The standard shall contain use case driven argumentation for safety sce- narios that can be used to build a safety case. (This should help the stack implementers in getting a certification, if they follow the standard.)
* Security
  + The standard shall specify data flow restrictions between its elements, and between applications.
  + The standard shall classify its elements with respect to security sensitivity (that is, functional clusters shall be marked if they handle sensitive data.)
  + The standard shall contain use case driven argumentation for security sce- narios that can be used to build a security case. (This should help the stack implementers in getting a certification, if they follow the standard.)
* Maintainability
  + It shall be possible in an efficient way to maintain AUTOSAR Adaptive Plat- form without preventing the introduction of new technologies (efficient in terms of effort on the modification of the standard).
  + The impact set of a change shall be available.
  + The standard shall be structured in a way that minimizes change impact.
  + It shall be possible to drop/deprecate elements of the standard.
  + It shall be easy to add new features/needs without breaking the maintain- ability or the need to redesign the software architecture. Easy means quick, with low effort, local changes only and no heavy side effects.
  + The maturity of parts of the standard shall be visible.
  + The process shall enforce an architectural impact analysis in a very early stage of the change process.
  + The process shall enforce minimizing changes, that is not adding similar functionality multiple times.
* Portability
  + Applications shall be portable between different stack implementations and different machines.
  + It shall be possible to scale the software architecture to the given project needs.
    1. **AUTOSAR Adaptive Platform Stack**
* Compatibility
  + An AUTOSAR Adaptive Platform stack implementation shall be capable to offer multiple versions of the same service.
  + An instance of an AUTOSAR Adaptive Platform stack implementation shall be able to co-exist with other instances on different machines, within the same vehicle.
* Usability
  + An AUTOSAR Adaptive Platform stack implementation shall explicitly doc- ument restrictions on the application development that go beyond the stan- dard.
* Maintainability
  + An AUTOSAR Adaptive Platform stack implementation shall be traceable to the contents of the standard.
  + An AUTOSAR Adaptive Platform stack implementation shall support multiple versions of the same service.
* Portability
  + An AUTOSAR Adaptive Platform stack shall be portable to a different custom hardware.
  + An AUTOSAR Adaptive Platform stack shall provide mechanisms to replace parts.
    1. **AUTOSAR Adaptive Application**
* Usability
  + *No Goal:* An application developer shall be able to supply custom implemen- tation for pre-defined platform functionality.
* Maintainability
  + An application shall explicitly state which parts of the standard it uses.
* Portability
  + An application entirely based on AUTOSAR Adaptive Platform (i.e. with- out custom extensions) shall be portable to another AUTOSAR Adaptive Platform stack of the same version without modifications to the application source code itself (source code compatibility).
  1. **Quality Scenarios**

There are currently no quality scenarios defined for the AUTOSAR Adaptive Platform.

1. **System Scope and Context**

This chapter provides an overview of the AUTOSAR Adaptive Platform system context by separating the AUTOSAR Adaptive Platform and its communication partners (e.g., external systems). Considering Figure [7.1](#_bookmark22), there are three categories of communica- tion partners for the AUTOSAR Adaptive Platform.



Adaptive Application

«use»

AUTOSAR Adaptive Platform

«use»

«use»

«use»

«use»

«use»



Operating System

«aapInternal» TCP/IP Stack

«aapInternal» Non-volatile Storage

«aapInternal»

Single-Process POSIX API

«aapInternal» Multi-Process

System Interface



Watchdog

«aapNativeInterf... Platform Health Management:: WatchdogInterface

**Figure 7.1:** **Overview of AUTOSAR Adaptive Platform and its context**

The AUTOSAR Adaptive Platform is conceptually a middleware. AUTOSAR Adaptive Platform provides services to [Adaptive Application](#_bookmark24)s beyond those available from the underlying operating system, drivers, and extensions (cf. Section [7.2](#_bookmark25)). Section

[7.3](#_bookmark31) describes the third category that are external systems communicating with (an

[Adaptive Application](#_bookmark24) via) AUTOSAR Adaptive Platform.

* 1. **Adaptive Application**

[Adaptive Application](#_bookmark24)s are built on the functionality provided by the AUTOSAR Adaptive Platform. They directly use the various interfaces provided by the individual building blocks of AUTOSAR Adaptive Platform described in more detail in chapter [9](#_bookmark55).

* 1. **Dependencies**
     1. **Crypto Provider**

Crypto Provider is a component that provides implementations of cryptographic routines and hash functions to the AUTOSAR Adaptive Platform.

* + 1. **Operating System**

The [Operating System](#_bookmark28) is the main component that AUTOSAR Adaptive Plat- form uses to provide its services. The [Operating System](#_bookmark28) controls processes and threads, and provides inter-process communication facilities. The [Operating Sys-](#_bookmark28) [tem](#_bookmark28) also provides access to network interfaces, protocols like TCP/IP, and access to non-volatile storage.

* + 1. **Watchdog**

The [Watchdog](#_bookmark30) is a component to control the hardware watchdog of the machine an AUTOSAR Adaptive Platform runs on.

* 1. **External Systems**



«flow»

«flow»

«flow»

«flow»

«flow»

Adaptive Application

Third-party Platform

Backend

AUTOSAR Classic Platform

AUTOSAR Adaptive Platform

Diagnostic Client

**Figure 7.2: Overview of the AUTOSAR Adaptive Platform and external systems**

The AUTOSAR Adaptive Platform supports applications that are operated in heteroge- neous environments. This section lists the external systems that AUTOSAR Adaptive Platform is intended to interface.

* + 1. **AUTOSAR Adaptive Application**

There may be many [Adaptive Application](#_bookmark24)s deployed in a vehicle on different Ma- chines. An [Adaptive Application](#_bookmark24) that does not run on the current instance of the AUTOSAR Adaptive Platform is therefore an external system to the AUTOSAR Adap- tive Platform. Such [Adaptive Application](#_bookmark24)s may exchange data such as sensor or status information by means of ara::com service interfaces.

* + 1. **AUTOSAR Classic Platform**

The [AUTOSAR Classic Platform](#_bookmark34) is the main platform for deeply-embedded appli- cations such as sensor/actor systems. [Adaptive Application](#_bookmark24)s may require ac- cess for example to sensor data provided by an [AUTOSAR Classic Platform](#_bookmark34) ECU and vice versa.

* + 1. **Third-party Platform**

Besides the both platforms (AUTOSAR Adaptive Platform and [AUTOSAR Classic](#_bookmark34) [Platform](#_bookmark34)) provided by AUTOSAR, there may be ECUs in a vehicle and other sys- tems that are built on different platforms that need to communicate with an [Adaptive](#_bookmark24) [Application](#_bookmark24) via AUTOSAR Adaptive Platform.

* + 1. **Diagnostic Client**

A [Diagnostic Client](#_bookmark37) uses the diagnostic services provided by the AUTOSAR Adaptive Platform.

* + 1. **Backend**

A [Backend](#_bookmark39) system provides Software Packages for download and controls the update process via [Update and Configuration Management](#_bookmark200).

1. **Solution Strategy**

The AUTOSAR Adaptive Platform is a standard for an automotive middleware. It is not a concrete implementation. The AUTOSAR Adaptive Platform standard leaves a certain degree of freedom to its implementers by defining requirements and software specifications that need to be fulfilled without specifying how.

* 1. **Architectural Approach**

To support the complex applications, while allowing maximum flexibility and scalabil- ity in processing distribution and compute resource allocations, AUTOSAR Adaptive Platform follows the concept of a service-oriented architecture (SOA). In a service- oriented architecture a system consists of a set of services, in which one may use another in turn, and applications that use one or more of the services depending on its needs. Often service-oriented architectures exhibit system-of-system characteris- tics, which AUTOSAR Adaptive Platform also has. A service, for instance, may reside on a local ECU that an application also runs, or it can be on a remote ECU, which is also running another instance of AP. The application code is the same in both cases - the communication infrastructure will take care of the difference providing transparent communication. Another way to look at this architecture is that of distributed comput- ing, communicating over some form of message passing. At large, all these represent the same concept. This message passing, communication-based architecture can also benefit from the rise of fast and high-bandwidth communication such as Ethernet.

* 1. **Decomposition Strategy**

The building blocks of the AUTOSAR Adaptive Platform architecture are refined step- by-step in this document according to the model depicted in figure [8.1](#_bookmark43). The top-level categories are chosen to give an overview from a users perspective what kind of func- tionality the AUTOSAR Adaptive Platform provides. A category contains one or more Functional Clusters. The Functional Clusters of the AUTOSAR Adaptive Platform are defined to group a specific coherent technical functionality. Functional Clusters themselves specify a set of interfaces and components to provide and real- ize that technical functionality. The building block view also contains information of the Functional Clusters interdependencies based on interfaces from other Func- tional Clusters they use. However, note that these interdependencies are recom- mendations rather than strict specifications because they would constrain implementa- tions.

**Category**

1..\*



**Functional Cluster**



|  |  |
| --- | --- |
| 1..\* |  |
| **Interface** | |

|  |  |
| --- | --- |
|  | 1..\* |
| **Component** | |

**Figure 8.1:** **Type model of building blocks**

* 1. **UML Profile**

The UML diagrams presented in this document use a UML profile to provide a more precise semantics of the elements and relationships. Table [8.1](#_bookmark45) provides an overview of the stereotypes in that profile and their semantics. The names of all stereotypes that are specific to the AUTOSAR architecture are prefixed with *aap* (short for AUTOSAR Architecture Profile) to make them easily distinguishable from standard UML stereo- types and keywords.

|  |  |  |
| --- | --- | --- |
| **Stereotype** | **Metaclass** | **Semantics** |
| aapInternal | Interface | Internal interfaces shall be used only by  Functional Clusters within the platform. In- ternal interfaces shall be realized by compo- nents that are part of the respective stack im- plementation (i.e., another Functional Cluster, additional middleware, drivers, or the operat- ing system). |
| aapPlatformExtension | Interface | Platform extension interfaces shall be used  only by Functional Clusters within the plat- form. Platform extension interfaces shall be realized either by third-party components (including application-level components) or components are part of the respective stack implementation (i.e., another Functional Clus- ter, additional middleware, drivers, or the op- erating system). |
| aapAPI | Interface | An interface of the public API of the plat-  form. Such interfaces may be used by Adap- tive Applications and other Functional Clus- ters within the platform. |
| aapNativeInterface | Interface | An interface defined in the respective pro-  gramming language of the stack implemen- tation. |

|  |  |  |
| --- | --- | --- |
| **Stereotype** | **Metaclass** | **Semantics** |
| aapPortInterface | Interface | An aapPortInterface relates to an ele-  ment in the Manifest using the Instance- Specifier pattern. Such interfaces are ei- ther defined in the respective programming language of the stack implementation or they are generated in that language by the stack tooling. The property Generated specifies if they are generated. |
| aapAraComServiceInterface | Interface | An ara::com service interface defined and  configured in the Manifest. The tooling of the stack implementation will generate stub and skeleton implementations of these in- terfaces in the respective programming lan- guage. |
| aapFunctionalCluster | Component | A functional cluster of the AUTOSAR Adap-  tive Platform. |
| aapServiceMethod | Operation | A method specified as part of an ara::com  service interface. |
| aapCallbackMethod | Operation | A method that acts as a callback. |
| aapServiceField | Attribute | A field specified as part of an ara::com ser-  vice interface. |
| aapServiceEvent | Attribute | An event specified as part of an ara::com ser-  vice interface. |
| aapProvidedPort | Usage, Realiza-  tion | Denotes that the underlying relationship is  configured in the Manifest using a provided port. |
| aapRequiredPort | Usage, Realiza-  tion | Denotes that the underlying relationship is  configured in the Manifest using a required port. |

**Table 8.1: Overview of Stereotypes**

* 1. **Technology**

### Implementation Language

C++ is the programming language of choice for the AUTOSAR Adaptive Platform and [Adaptive Application](#_bookmark24)s. C++ was chosen due to its safer programming model (compared to C) and availability of certified compilers that produce highly optimized machine code. Such properties are especially important for safety- and performance- critical, real-time applications (such as typical [Adaptive Application](#_bookmark24)s) where C++ has become more and more popular in the software industry and academics.

### Parallel Processing

Although the design for AUTOSAR Adaptive Platform as a service-oriented architecture inherently leverages parallel processing, the advancement of (heterogeneous) many-

core processors offers additional opportunities. The AUTOSAR Adaptive Platform is designed to scale its functionality and performance as (heterogeneous) many-core technologies advance. Hardware and platform interface specifications are one part of that equation. However, advancements in operating system and hypervisor tech- nologies as well as development tools (for example automatic parallelization) are also crucial and are to be fulfilled by AUTOSAR Adaptive Platform providers, the software industry, and academics.

## Design Principles

The architecture of the AUTOSAR Adaptive Platform is based on several design prin- ciples that are outlined below.

### Leveraging existing standards

AUTOSAR Adaptive Platform aims to leverage existing standards and specifications wherever possible. For example, AUTOSAR Adaptive Platform ist built on the existing and open C++ standard (cf. Section [8.4.1](#_bookmark47)) to facilitate a faster development of the AUTOSAR Adaptive Platform itself and benefiting from the eco-systems of such stan- dards. It is, therefore, a critical focus in developing the AUTOSAR Adaptive Platform specification not to casually introduce a new replacement functionality that an existing standard already offers. For instance, no new interfaces are casually introduced just because an existing standard provides the functionality required but the interface is superficially hard to understand.

### SOLID principles

The SOLID principles [[7](#_bookmark329)] are a central part of the design principles of AUTOSAR. While these five principles are all valid, only the Single-responsibility Principle, the Interface Segregation Principle and the Dependency Inversion Principle are relevant on the ab- straction level of this document. Therefore, they are elaborated in the following.

* + - 1. **Single-responsibility Principle**

The single-responsibility principle (SRP,SWEBOK3) [[7](#_bookmark329)] states that a component or class should be responsible for a single part of the overall functionality provided by the soft- ware. That responsibility should be encapsulated by the component or class. The services provided by the component or class (via its interface(s)) should be closely aligned with its responsibility.

The single-responsibility principle minimizes the reasons (i.e. a change to the single responsibility) that require a change to its interface. Thus, it minimizes impact on clients of such an interface and leads to a more maintainable architecture (or code).

* + - 1. **Interface Segregation Principle**

The interface segregation principle (ISP) [[7](#_bookmark329)], [[8](#_bookmark330)] states that clients should not be forced to depend on methods that they don’t use. As a consequence of the interface segre- gation principle, interfaces should be split up to reflect different roles of clients.

Similar to the single-responsibility principle, the segregation of interfaces reduce the impact of a change to an interface to the clients and suppliers of an segregated inter- face.

* + - 1. **Dependency Inversion Principle**

The dependency inversion principle (DIP) [[7](#_bookmark329)], [[8](#_bookmark330)] states that high-level building blocks should not depend on low-level building blocks. Both should depend on abstractions (e.g. interfaces). Furthermore, the dependency inversion principle states that abstrac- tions (e.g. interfaces) should not depend on details. Details (e.g. a concrete imple- mentation) should depend on abstractions.

The dependency inversion principle results in a decoupling of the implementations of building blocks. This is important to scale implementation efforts (cf. Section [5.2](#_bookmark14)) and to perform proper integration tests.

### Acyclic Dependencies Principle

The acyclic dependencies principle (ADP) [[7](#_bookmark329)], [[8](#_bookmark330)] states that dependencies between building blocks should form a directed acyclic graph.

The acyclic dependencies principle helps to identify participating building blocks and reason about error propagation and freedom from interference. In general, it also re- duces the extend of building blocks to consider during activities such as test, build and deployment.

## Deployment

The AUTOSAR Adaptive Platform supports the incremental deployment of applica- tions, where resources and communications are managed dynamically to reduce the effort for software development and integration, enabling short iteration cycles. Incre- mental deployment also supports explorative software development phases. For prod- uct delivery, the AUTOSAR Adaptive Platform allows the system integrator to carefully

limit dynamic behavior to reduce the risk of unwanted or adverse effects allowing safety qualification. Dynamic behavior of an application will be limited by constraints stated in the Execution Manifest (cf. Section [13.8](#_bookmark308)), for example, dynamic allocation of resources and communication paths are only possible in defined ways, within config- ured ranges. Implementations of an AUTOSAR Adaptive Platform may further remove dynamic capabilities from the software configuration for production use. Examples of reduced behavioral dynamics might be:

* Pre-determination of the service discovery process
* Restriction of dynamic memory allocation to the startup phase only
* Fair scheduling policy in addition to priority-based scheduling
* Fixed allocation of processes to CPU cores
* Access to pre-existing files in the file-system only
* Constraints for AUTOSAR Adaptive Platform API usage by applications
* Execution of authenticated code only

## Verification and Validation

The AUTOSAR Adaptive Platform standard uses a dedicated implementation of the standard (AUTOSAR Adaptive Platform Demonstrator) to validate the requirements and to verify the (still abstract) software design imposed by the individual software specifications.

# Building Block View

This chapter provides an overview of the static structure of the AUTOSAR Adaptive Platform by describing the high-level building blocks and their inter-dependencies. Please note that the use of interfaces between Functional Clusters in the AU- TOSAR Adaptive Platform is currently not standardized. Some aspects, for example, access management, are also not yet fully incorporated and standardized in all Func- tional Clusters.

## Overview

Figure [9.1](#_bookmark57) provides an overview of the different categories of building blocks available in the AUTOSAR Adaptive Platform. The categories are explained in more detail in the subsequent sections.

AUTOSAR Adaptive Platform

Diagnostics

Configuration

Safety

Security

Storage

Communication

Runtime

**Figure 9.1:** **Overview of the AUTOSAR Adaptive Platform and its building block cate- gories**

### Description pattern

The description of all building blocks (FunctionalClusters) in this section uses the same pattern. Each FunctionalCluster is described in a separate section of the document. Such a section starts with an overview of the responsibilities of the FunctionalCluster followed by a sub-section called "Defined interfaces". The sub- section "Defined interfaces" lists all architectural interfaces specified in the namespace of the FunctionalCluster. Each interface is detailed in a separate table.

The subsection "Provided interfaces" then lists all interfaces provided by the Func- tionalCluster (i.e. it is fully implemented by the FunctionalCluster) to other FunctionalClusters.

The last subsection "Required interfaces" lists all interfaces required by the Func- tionalCluster from other FunctionalClusters and external components like the operating system.

## Runtime



«aapFunctionalCluster»

Execution Management

daemon-based



«aapFunctionalCluster»

State Management

daemon-based



«aapFunctionalCluster» Log and Trace



«aapFunctionalCluster» Core



«aapFunctionalCluster» Operating System Interface

**Figure 9.2: Functional Clusters in category Runtime**

### Execution Management

|  |  |
| --- | --- |
| ***Name:*** | Execution Management |
| ***Short Name:*** | exec |
| ***Category:*** | Runtime |
| ***Daemon-based:*** | Yes |
| ***Responsibilities:*** | [Execution Management](#_bookmark61) is responsible to control Processes of the AUTOSAR Adaptive Platform and [Adaptive Application](#_bookmark24)s. That is, it starts, configures, and stops Processes as configured in Function Group States using interfaces of the [Operating System](#_bookmark28). The [Operating System](#_bookmark28) is responsible for runtime scheduling of those Processes. The configuration of Processes that [Execution Management](#_bookmark61) performs includes limiting their resource consumption (CPU time, memory) using Resource Groups provided by the [Operating](#_bookmark28) [System](#_bookmark28).  [Execution Management](#_bookmark61) is the entry point of AUTOSAR Adaptive Platform and is started by the [Operating System](#_bookmark28) during system boot. [Execution Management](#_bookmark61) then controls the startup and shutdown of the AUTOSAR Adaptive Platform (see use cases [Start Adaptive Platform](#_bookmark242) and [Shutdown Adaptive Platform](#_bookmark243) for details). [Execution Management](#_bookmark61) optionally supports authenticated startup where it maintains the chain of trust when starting from a Trust Anchor.  During authenticated startup [Execution Management](#_bookmark61) validates the authenticity and integrity of Processes and shall prevent their execution if violations are detected. Through these mechanisms, a trusted platform can be established. |

* + - 1. **Defined interfaces**

The interfaces of [Execution Management](#_bookmark61) are categorized into interfaces for state reporting (see Section [9.2.1.1.1](#_bookmark62)), interfaces for the deterministic execution (see Section [9.2.1.1.2](#_bookmark65)), and interfaces for [State Management](#_bookmark76) (see Section [9.2.1.1.3](#_bookmark69)).

* + - * 1. **Interfaces for state reporting**

All processes started by [Execution Management](#_bookmark61) (i.e. all processes of the AU- TOSAR Adaptive Platform and all processes of Adaptive Applications) shall report their execution state back to [Execution Management](#_bookmark61) via the [ExecutionClient](#_bookmark64) interface (cf. Figure [9.3](#_bookmark63)).



Adaptive Application

«use»



«aapFunctionalCluster»

Execution Management daemon-based

+ ReportExecutionState(ExecutionState): Result

«aapAPI,aapNativeInterface» ExecutionClient

**Figure 9.3:** **Interfaces for state reporting**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | ExecutionClient | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides functionality for a Process to report its execution state to [Execution](#_bookmark61) [Management](#_bookmark61). | |
| ***Operations:*** | ReportExecutionState | Report the internal state of a Process to [Execution](#_bookmark61) [Management](#_bookmark61). |

* + - * 1. **Interfaces for deterministic execution**

The [DeterministicClient](#_bookmark67) API (cf. Figure [9.4](#_bookmark66)) provides operations to perform determin- istic execution of tasks.



Adaptive Application

«use»

«use»

«aapAPI,aapNativeInterface» DeterministicClient

«aapAPI,aapNativeInterface» WorkerThread

«aapAPI,aapNativeInterface» WorkerRunnable

+ GetActivationTime()

+ GetNextActivationTime()

+ GetRandom()

+ RunWorkerPool()

+ SetRandomSeed()

+ WaitForActivation()

+ GetRandom()

+ Run(ValueType, WorkerThread)

«use»

daemon-based

«aapFunctionalCluster» Execution Management

**Figure 9.4:** **Interfaces for deterministic execution**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | DeterministicClient | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides the functionality for an application to run a cyclic deterministic execution. Each modelled Process which needs support for cyclic deterministic execution has to instantiate this interface. | |
| ***Operations:*** | GetActivationTime | Get the timestamp of the activation point. |
| GetNextActivationTime | Get the timestamp of the next activation point. |
| GetRandom | Returns a deterministic sequence of random numbers. |
| RunWorkerPool | Runs tasks in a deterministic worker pool. |
| SetRandomSeed | Seed the random number generator used for redundantly executed deterministic clients. |
| WaitForActivation | Blocks and returns with a process control value when the next activation is triggered by the runtime. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | WorkerRunnable | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface is used to implement worker runnables for the [DeterministicClient](#_bookmark67). | |
| ***Operations:*** | Run | Runs the deterministic client worker runnable. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | WorkerThread | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface is used to implement worker threads for the [DeterministicClient](#_bookmark67). | |
| ***Operations:*** | GetRandom | Returns a deterministic pseudo-random number which is unique for each container element in the worker pool. |

* + - * 1. **Interfaces for State Management**

The [StateClient](#_bookmark71) API (cf. Figure [9.5](#_bookmark70)) provides operations to control [FunctionGroup](#_bookmark74)s and their [FunctionGroupState](#_bookmark73)s.



«aapFunctionalCluster»

State Management

daemon-based

«aapAPI,aapPortInterface» FunctionGroup

«aapAPI,aapPortInterface» FunctionGroupState



«use»

«use»

«aapProvidedPort»

«use»

«aapProvidedPort»

«aapAPI,aapNativeInterface» StateClient

+ GetExecutionError()

+ GetInitialMachineStateTransitionResult()

+ SetState(FunctionGroupState): Future

+ StateClient(function)

«aapRequiredPort»

«aapRequiredPort»

«aapFunctionalCluster» Execution Management

daemon-based

**Figure 9.5:** **Interfaces for State Management**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | StateClient | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides the functionality to request [FunctionGroupState](#_bookmark73) transitions and to perform error detection and error handling. | |
| ***Operations:*** | GetExecutionError | Returns the execution error which changed the given [FunctionGroup](#_bookmark74) to an undefined [FunctionGroupState](#_bookmark73). |
| GetInitialMachineStateTransitionResult | Retrieve the result of Machine State initial transition to Startup state. |
| SetState | Request a [FunctionGroupState](#_bookmark73) transition for a single [FunctionGroup](#_bookmark74). |
| StateClient | Constructor of this interface. It requires a callback to be invoked if a [FunctionGroup](#_bookmark74) changes its state unexpectedly to an undefined [FunctionGroupState](#_bookmark73), i.e. without previous request by [SetState()](#_bookmark72). The affected [FunctionGroup](#_bookmark74) is provided as an argument to the callback. |

|  |  |
| --- | --- |
| ***Name:*** | FunctionGroupState |
| ***Technology:*** | Port interface |
| ***Generated:*** | No |
| ***Meta-model interface type:*** | ModeDeclaration |
| ***Usage:*** | Public API |
| ***Description:*** | Represents a Function Group State defined in the Manifest. |

|  |  |
| --- | --- |
| ***Name:*** | FunctionGroup |
| ***Technology:*** | Port interface |
| ***Generated:*** | No |
| ***Meta-model interface type:*** | ModeDeclarationGroup |
| ***Usage:*** | Public API |
| ***Description:*** | Represents a Function Group defined in the Manifest. |

* + - 1. **Provided interfaces**



«aapFunctionalClust...

State Management

daemon-based



«aapFunctionalClust...

Platform Health Management

daemon-based



«aapFunctionalClust... Diagnostic Management

daemon-based



«aapFunctionalClust... Update and Configuration Management

daemon-based



«aapFunctionalClust...

Time Synchronization

daemon-based

«use»



daemon-based

«aapFunctionalCluster» Execution Management

+ ReportExecutionState(ExecutionState): Result

«aapAPI,aapNativeInterface» ExecutionClient

«use»

«use»

«use»

«use»

**Figure 9.6: Users of the ExecutionClient interface**



«aapFunctionalCluster»

State Management

daemon-based

«aapAPI,aapPortInterface» FunctionGroup

«aapAPI,aapPortInterface» FunctionGroupState



«use»

«use»

«aapProvidedPort»

«use»

«aapProvidedPort»

«aapAPI,aapNativeInterface» StateClient

+ GetExecutionError()

+ GetInitialMachineStateTransitionResult()

+ SetState(FunctionGroupState): Future

+ StateClient(function)

«aapRequiredPort»

«aapRequiredPort»

«aapFunctionalCluster» Execution Management

daemon-based

**Figure 9.7: Users of the State Management interfaces**

|  |  |
| --- | --- |
| ***Interface*** | ***Requiring functional clusters*** |
| [Execution Management](#_bookmark61)::[ExecutionClient](#_bookmark64) | [Diagnostic Management](#_bookmark214) |
| [Platform Health Management](#_bookmark186) |
| [State Management](#_bookmark76) |
| [Time Synchronization](#_bookmark115) |

*Δ*

|  |  |
| --- | --- |
| ***Interface*** | ***Requiring functional clusters*** |
|  | [Update and Configuration Management](#_bookmark200) |
| [Execution Management](#_bookmark61)::[FunctionGroup](#_bookmark74) | [State Management](#_bookmark76) |
| [Execution Management](#_bookmark61)::[FunctionGroupState](#_bookmark73) | [State Management](#_bookmark76) |
| [Execution Management](#_bookmark61)::[StateClient](#_bookmark71) | [State Management](#_bookmark76) |

**Table 9.1: Interfaces provided by Execution Management to other Functional Clusters**

* + - 1. **Required interfaces**

«aapFunctionalCluster» Execution Management

daemon-based

«use»

«use»

«aapRequiredPort»



«use»

«aapRequiredPort»

«use»



«aapInternal»

Multi-Process System Interface

+ SetProcessConfiguration()

+ StartProcess()

+ TerminateProcess()

Operating System



«aapAPI,aapNativeInterf...

Logger

+ IsEnabled()

+ Log(MsgId, Params)

+ LogDebug(): LogStream

+ LogError(): LogStream

+ LogFatal(): LogStream

+ LogInfo(): LogStream

+ LogVerbose(): LogStream

+ LogWarn(): LogStream

+ WithLevel(): LogStream

«aapFunctionalClust...

Log and Trace



«aapAPI,aapPortInterface» SynchronizedTimeBaseConsumer

+ GetCurrentTime()

+ GetRateDeviation()

+ GetTimeWithStatus(): SynchronizedTimeBaseStatus

+ RegisterStatusChangeNotifier()

+ RegisterSynchronizationStateChangeNotifier()

+ RegisterTimeLeapNotifier()

+ RegisterTimePrecisionMeasurementNotifier()

+ RegisterTimeValidationNotification()

+ UnregisterStatusChangeNotifier()

+ UnregisterSynchronizationStateChangeNotifier()

+ UnregisterTimeLeapNotifier()

+ UnregisterTimePrecisionMeasurementNotifier()

+ UnregisterTimeValidationNotification()

«aapFunctionalCluster»

Time Synchronization

daemon-based



«aapAPI,aapPortInterface» SupervisedEntity

+ ReportCheckpoint()

«aapFunctionalClust...

Platform Health Management

daemon-based

**Figure 9.8: Interfaces required by Execution Management**

|  |  |
| --- | --- |
| ***Interface*** | ***Purpose*** |
| Multi-Process System Interface | [Execution Management](#_bookmark61) shall use this interface to start, configure and control os-level processes. |
| [Adaptive Intrusion Detection System Manager](#_bookmark178):: [EventReporter](#_bookmark179) | [Execution Management](#_bookmark61) shall use this interface to report security events (e.g., verification failure of an executable). |
| [Execution Management](#_bookmark61)::[WorkerRunnable](#_bookmark68) | [Execution Management](#_bookmark61) shall use this interface to execute [WorkerRunnable](#_bookmark68)s using its [DeterministicClient](#_bookmark67) implementation. |
| [Log and Trace](#_bookmark90)::[Logger](#_bookmark91) | [Execution Management](#_bookmark61) shall use this interface to log standardized messages. |
| [Persistency](#_bookmark124)::[KeyValueStorage](#_bookmark141) | [Execution Management](#_bookmark61) should use this interface to read/write persistent data. |
| [Platform Health Management](#_bookmark186)::[SupervisedEntity](#_bookmark189) | [Execution Management](#_bookmark61) shall use this interface to enable supervision of its process(es) by [Platform Health Management](#_bookmark186). |

*Δ*

|  |  |
| --- | --- |
| ***Interface*** | ***Purpose*** |
| [Registry](#_bookmark210)::[ManifestAccessor](#_bookmark211) | [Execution Management](#_bookmark61) shall use this interface to read the configuration of its [DeterministcClient](#_bookmark67) and information about Function Groups and Processes from the Manifests. |
| [Time Synchronization](#_bookmark115)::[SynchronizedTimeBase](#_bookmark120) [Consumer](#_bookmark120) | The [DeterministicClient](#_bookmark67) implementation in [Execution Management](#_bookmark61)  should use this interface to synchronize execution of [DeterministicClient](#_bookmark67)s. |

**Table 9.2: Interfaces required by Execution Management**

### State Management

|  |  |
| --- | --- |
| ***Name:*** | State Management |
| ***Short Name:*** | sm |
| ***Category:*** | Runtime |
| ***Daemon-based:*** | Yes |
| ***Responsibilities:*** | [State Management](#_bookmark76) determines the desired target state of the AUTOSAR Runtime for Adaptive Applications based on various application-specific inputs. That target state is the set of active Function Group States of all Function Groups running on the AUTOSAR Runtime for Adaptive Applications. [State Management](#_bookmark76) delegates to [Execution Management](#_bookmark61) to switch the individual Function Groups to the respective Function Group States.  [State Management](#_bookmark76) is a unique component in the AUTOSAR Adaptive Platform because it is not part of an AUTOSAR Adaptive Platform stack. The logic of [State Management](#_bookmark76) currently needs to be implemented as application-specific code and then configured and integrated with an AUTOSAR Adaptive Platform stack. |

* + - 1. **Defined interfaces**

The interfaces of [State Management](#_bookmark76) are categorized into interfaces for triggering state changes (see Section [9.2.2.1.1](#_bookmark77)), interfaces for diagnostic reset (see Section [9.2.2.1.2](#_bookmark79)), interfaces for requesting a Power Mode (see Section [9.2.2.1.3](#_bookmark83)), and inter- faces for interaction with [Update and Configuration Management](#_bookmark200) (see Section [9.2.2.1.4](#_bookmark87)).

* + - * 1. **Interfaces for triggering state changes**

[State Management](#_bookmark76) provides several interface blueprints to get and set its internal state (cf. Figure [9.9](#_bookmark78)).

«use»

«aapRequiredPort»

«aapAccessControlled, aapServiceFie...

+ Notifier

«aapAraComServiceInterface,aapAPI» TriggerOut\_{StateGroup}

«use»

«aapRequiredPort»

«aapAccessControlled, aapServiceField»

+ Notifier

+ Trigger

«aapAraComServiceInterface,aapAPI» TriggerInOut\_{StateGroup}



«use»

«aapRequiredPort»

«aapAccessControlled, aapServiceFie...

+ Trigger

«aapAraComServiceInterface,aapAPI» TriggerIn\_{StateGroup}

Adaptive Application

«aapProvidedPort»

«aapProvidedPort»

«aapProvidedPort»



«aapFunctionalCluster»

State Management

daemon-based

**Figure 9.9:** **Interfaces for triggering state changes**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | TriggerIn\_{StateGroup} | |
| ***Technology:*** | ara::com service interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface is to be used by [Adaptive Application](#_bookmark24)s to trigger [State Management](#_bookmark76) to change its internal state. | |
| ***Fields:*** | Trigger | A value to be evaluated by [State Management](#_bookmark76) in a project-specific way. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | TriggerOut\_{StateGroup} | |
| ***Technology:*** | ara::com service interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface is to be used by [Adaptive Application](#_bookmark24)s to be informed when [State](#_bookmark76) [Management](#_bookmark76) has changed its internal state. | |
| ***Fields:*** | Notifier | To be set by [State Management](#_bookmark76) in a project-specific way to inform [Adaptive Application](#_bookmark24)s about changes within [State Management](#_bookmark76). |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | TriggerInOut\_{StateGroup} | |
| ***Technology:*** | ara::com service interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface is to be used by [Adaptive Application](#_bookmark24)s to trigger [State Management](#_bookmark76) to change its internal state and to get information when it is carried out. | |
| ***Fields:*** | Notifier | To be set by [State Management](#_bookmark76) in a project-specific way to inform [Adaptive Application](#_bookmark24)s about changes within [State Management](#_bookmark76). |
| Trigger | A value to be evaluated by [State Management](#_bookmark76) in a project-specific way. |

* + - * 1. **Interfaces for requesting a diagnostic reset**

The [DiagnosticReset](#_bookmark80) interface propagates a diagnostic reset request ([DiagnosticRe-](#_bookmark82) [set::message()](#_bookmark82)) to all affected Processes. These Processes then shall answer the diagnostic reset request by calling [DiagnosticReset::event()](#_bookmark81).



Adaptive Application



«aapFunctionalCluster»

Diagnostic Management daemon-based

«use»

«aapRequiredPort»

«use»

«aapRequiredPort»

«aapAPI,aapAraComServiceInterface» DiagnosticReset

«aapAccessControlled, aapServiceMethod»

+ event(DiagnosticResetRespMsg\*)

+ message(DiagnosticResetMsg\*)

«aapProvidedPort»



«aapFunctionalCluster»

State Management

daemon-based

**Figure 9.10: Interface for handling a diagnostic reset**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | DiagnosticReset | |
| ***Technology:*** | ara::com service interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides functionality to handle diagnostic reset requests. | |
| ***Operations:*** | event | All Processes which got a diagnostic reset request shall call this method to provide an answer to [State](#_bookmark76) [Management](#_bookmark76). |
| message | Sends a diagnostic reset message to all affected  Processes. |

* + - * 1. **Interfaces for requesting a Power Mode**

The [PowerMode](#_bookmark84) interface propagates a diagnostic Power Mode request ([Power-](#_bookmark86) [Mode::message()](#_bookmark86)) to all running Processes. These Processes then shall answer the Power Mode request by calling [PowerMode::event()](#_bookmark85).



Adaptive Application

«use»

«aapRequiredPort»

«aapAccessControlled, aapService...

+ event(PowerModeRespMsg\*)

+ message(PowerModeMsg\*)

«aapAPI,aapAraComServiceInterfa...

PowerMode

«aapProvidedPort»



«aapFunctionalCluster»

State Management daemon-based

**Figure 9.11: Interface for handling a Power Mode request**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | PowerMode | |
| ***Technology:*** | ara::com service interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides functionality to handle Power Mode requests. | |
| ***Operations:*** | event | All Processes which have received a Power Mode request shall call this method to provide an answer to [State Management](#_bookmark76). |
| message | Sends a Power Mode request to all running  Processes. |

* + - * 1. **Interfaces for interaction with Update and Configuration Management**



UCM Subordinate



«use»

«aapRequiredPort»

«aapProvidedPort»

«aapAccessControlled, aapServiceMe...

+ PrepareRollback()

+ PrepareUpdate()

+ RequestUpdateSession()

+ ResetMachine()

+ StopUpdateSession()

+ VerifyUpdate()

«aapAraComServiceInterface,aapInte...

UpdateRequest

«aapFunctionalCluster»

State Management daemon-based

**Figure 9.12: Interface for software update handling**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | UpdateRequest | |
| ***Technology:*** | ara::com service interface | |
| ***Usage:*** | Internal | |
| ***Description:*** | This interface is intended to be used by [Update and Configuration Management](#_bookmark200) to interact with [State Management](#_bookmark76) to perform updates (including installation and removal) of Software Clusters. | |
| ***Operations:*** | PrepareRollback | Prepares the affected Function Groups for a rollback. |
| PrepareUpdate | Prepares the affected Function Groups for an update. |
| RequestUpdateSession | Requests an update session. [State Management](#_bookmark76) might decline this request when the Machine is not in a state to be updated. |
| ResetMachine | Requests an orderly reset of the Machine. Before the reset is performed all information within the Machine shall be persisted. |
| StopUpdateSession | Ends an update session. |

*q*

*Δ*

|  |  |  |
| --- | --- | --- |
|  | VerifyUpdate | Verifies the affected Function Groups after an update. |

* + - 1. **Provided interfaces**



UCM Subordinate



«aapFunctionalCluster»

Diagnostic Management

daemon-based



«aapFunctionalCluster»

Platform Health Management daemon-based



«use»

«aapRequiredPort»

«use»

«aapRequiredPort»

«use»

«aapRequiredPort»

«use»

«aapPortInterface,aapAPI» EcuResetRequest

+ EnableRapidShutdown()

+ ExecuteReset()

+ Offer()

+ RequestReset()

+ StopOffer()

«aapProvidedPort»

«aapProvidedPort»

«aapProvidedPort»

«aapFunctionalCluster» State Management

daemon-based

+ GetGlobalSupervisionStatus()

+ Offer()

+ StopOffer()

«aapCallbackMethod»

+ RecoveryHandler()

«aapPortInterface,aapAPI» RecoveryAction

«aapAccessControlled, aapServiceMe...

+ PrepareRollback()

+ PrepareUpdate()

+ RequestUpdateSession()

+ ResetMachine()

+ StopUpdateSession()

+ VerifyUpdate()

«aapAraComServiceInterface,aapInte...

UpdateRequest

**Figure 9.13: Users of the State Management interfaces**

«aapAccessControlled, aapServic...

+ event(DiagnosticResetRespMsg\*)

+ message(DiagnosticResetMsg\*)

«aapAPI,aapAraComServiceInter...

DiagnosticReset

|  |  |
| --- | --- |
| ***Interface*** | ***Requiring functional clusters*** |
| [Diagnostic Management](#_bookmark214)::[Communication](#_bookmark225) [Control](#_bookmark225) | [Diagnostic Management](#_bookmark214) |
| [Diagnostic Management](#_bookmark214)::[EcuResetRequest](#_bookmark226) | [Diagnostic Management](#_bookmark214) |
| [Platform Health Management](#_bookmark186)::[RecoveryAction](#_bookmark192) | [Platform Health Management](#_bookmark186) |
| [State Management](#_bookmark76)::[DiagnosticReset](#_bookmark80) | [Diagnostic Management](#_bookmark214) |
| [State Management](#_bookmark76)::[UpdateRequest](#_bookmark88) | [Update and Configuration Management](#_bookmark200) |

**Table 9.3: Interfaces provided by State Management to other Functional Clusters**

* + - 1. **Required interfaces**

daemon-based

«aapFunctionalCluster» State Management

«use»

«aapRequiredPort»

«use»

«aapRequiredPort»

«use»

«aapAPI,aapPortInterface» SupervisedEntity

+ ReportCheckpoint()

«aapAraComServiceInterface,aapAPI» NetworkState\_{NetworkHandle}

«aapAccessControlled, aapServiceField»

+ NetworkCurrentState

+ NetworkRequestedState

«aapPortInterface,aapAPI» CommunicationControl

+ CommCtrlRequest()

+ Offer()

+ StopOffer()

«aapAPI,aapPortInterface» KeyValueStorage

+ DiscardPendingChanges()

+ GetAllKeys()

+ GetValue()

+ KeyExists()

+ RecoverKey()

+ RemoveAllKeys()

+ RemoveKey()

+ ResetKey()

+ SetValue()

+ SyncToStorage()

«aapFunctionalCluster»

Platform Health Management daemon-based

«aapProvidedPort»

«aapFunctionalCluster» Network Management

«use»

«aapRequiredPort»

«aapFunctionalClust...



«aapFunctionalClust... Diagnostic Management daemon-based

Persistency

**Figure 9.14: Interfaces required by State Management**



«aapFunctionalCluster»

State Management

daemon-based

«use»



«use»

«aapProvidedPort»



«aapRequiredPort»

«aapRequiredPort»

«aapFunctionalCluster» Execution Management

daemon-based

+ GetExecutionError()

+ GetInitialMachineStateTransitionResult()

+ SetState(FunctionGroupState): Future

+ StateClient(function)

«aapAPI,aapNativeInterface» Execution Management::StateClient

«use»

«aapProvidedPort»

«aapAPI,aapPortInterface» Execution Management:: FunctionGroup

«aapAPI,aapPortInterface» Execution Management:: FunctionGroupState

**Figure 9.15: Interfaces of Execution Management required by State Management**

|  |  |
| --- | --- |
| ***Interface*** | ***Purpose*** |
| [Execution Management](#_bookmark61)::[ExecutionClient](#_bookmark64) | This interface shall be used to report the state of the [State](#_bookmark76) [Management](#_bookmark76) process(es). |
| [Execution Management](#_bookmark61)::[FunctionGroup](#_bookmark74) | This interface shall be used to request [FunctionGroupState](#_bookmark73) transitions and to check their status. |
| [Execution Management](#_bookmark61)::[FunctionGroupState](#_bookmark73) | This interface shall be used to request [FunctionGroupState](#_bookmark73) transitions and to check their status. |
| [Execution Management](#_bookmark61)::[StateClient](#_bookmark71) | This interface shall be used to request [FunctionGroupState](#_bookmark73) transitions. |
| [Log and Trace](#_bookmark90)::[Logger](#_bookmark91) | [State Management](#_bookmark76) shall use this interface to log standardized messages. |
| [Network Management](#_bookmark112)::[NetworkState\_{Network](#_bookmark113) [Handle}](#_bookmark113) | This interface shall be used to retrieve information about the network status of a NetworkHandle. |
| [Persistency](#_bookmark124)::[KeyValueStorageOperations](#_bookmark138) | This interface should be used to persist information (e.g. update session). |
| [Persistency](#_bookmark124)::[KeyValueStorage](#_bookmark141) | This interface should be used to persist information (e.g. update session). |
| [Platform Health Management](#_bookmark186)::[SupervisedEntity](#_bookmark189) | [State Management](#_bookmark76) shall use this interface to enable supervision of its process(es) by [Platform Health Management](#_bookmark186). |

**Table 9.4: Interfaces required by State Management**

### Log and Trace

|  |  |
| --- | --- |
| ***Name:*** | Log and Trace |
| ***Short Name:*** | log |
| ***Category:*** | Runtime |
| ***Daemon-based:*** | No |
| ***Responsibilities:*** | [Log and Trace](#_bookmark90) provides functionality to build and log messages of different severity. An [Adaptive Application](#_bookmark24) can be configured to forward log messages to various sinks, for example to a network, a serial bus, the console, and to non-volatile storage. |

* + - 1. **Defined interfaces**

The entry point to the Log and Trace framework is the [CreateLogger()](#_bookmark93) operation that constructs a new [Logger](#_bookmark91) context. Afterwards, new messages can be constructed using the [LogStream](#_bookmark94) that is returned by the operations in [Logger](#_bookmark91), for example [LogInfo](#_bookmark92) [()](#_bookmark92).



Adaptive Application

«use»

«use»

«use»



«aapFunctionalCluster» Log and Trace

+ Flush(): void

+ operator<<(): void

+ WithLocation(): void

+ Arg()

+ BinFormat()

+ CreateLogger(): Logger

+ HexFormat()

+ remoteClientState()

+ IsEnabled()

+ Log(MsgId, Params)

+ LogDebug(): LogStream

+ LogError(): LogStream

+ LogFatal(): LogStream

+ LogInfo(): LogStream

+ LogVerbose(): LogStream

+ LogWarn(): LogStream

+ WithLevel(): LogStream

«aapAPI,aapNativeInterface» Log and Trace::LogStream

«aapAPI,aapNativeInt...

Log and Trace

«aapAPI,aapNativeInterf... Log and Trace::Logger

**Figure 9.16: Interfaces of Log and Trace**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | Logger | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface represents a logger context. The logging framework defines contexts which can be seen as logger instances within one application process or process scope. A context will be automatically registered against the Logging back-end during creation phase, as well as automatically de-registered during process shutdown phase. | |
| ***Operations:*** | IsEnabled | Check if the provided log level is enabled in the current configured log level. |
| Log | Logs a message modeled in the Manifest. |
| LogDebug | Creates a [LogStream](#_bookmark94) object with Debug severity. |
| LogError | Creates a [LogStream](#_bookmark94) object with Error severity. |
| LogFatal | Creates a [LogStream](#_bookmark94) object with Fatal severity. |
| LogInfo | Creates a [LogStream](#_bookmark94) object with Info severity. |
| LogVerbose | Creates a [LogStream](#_bookmark94) object with Verbose severity. |
| LogWarn | Creates a [LogStream](#_bookmark94) object with Warn severity. |
| WithLevel | Creates a [LogStream](#_bookmark94) object with the provided log level. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | LogOperations | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides access to the logging framework and utility operations to control the format of value printed to the log output. | |
| ***Operations:*** | Arg | Create a wrapper object. The wrapper object holds a value and an optional name and unit of the value. |
| BinFormat | Conversion of an integer into a binary value. Negatives are represented in 2’s complement. The number of represented digits depends on the overloaded parameter type length. |
| CreateLogger | Creates a [Logger](#_bookmark91) object, holding the context which is registered in the logging framework. |

*q*

*Δ*

|  |  |  |
| --- | --- | --- |
|  | HexFormat | Conversion of an integer into a hexadecimal value. Negatives are represented in 2’s complement. The number of represented digits depends on the overloaded parameter type length. |
| remoteClientState | Fetches the connection state from the DLT back-end of a possibly available remote client. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | LogStream | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides functionality to construct a single log message by appending data using stream operators. | |
| ***Operations:*** | Flush | Sends out the current log buffer and initiates a new message stream. Calling this operation is only necessary if the [LogStream](#_bookmark94) is intended to be reused within the same scope. |
| WithLocation | Add source file location into the message. |
| operator<< | Writes a value into the log message. Several overloads exist to control the output format. |

* + - 1. **Provided interfaces**



«aapFunctionalClu... Execution Management daemon-based



«aapFunctionalClust...

Cryptography daemon-based

«aapFunctionalClu...

State Management daemon-based



«aapFunctionalClust... Identity and Access Management



«aapFunctionalCluster» Communication Management

«aapFunctionalClus...

Persistency

«use»



«aapFunctionalClust... Diagnostic Management daemon-based



«aapFunctionalClust...

Time Synchronization daemon-based



«aapFunctionalClu... Network Management

«use»

«use»

«use»

«use» «use»

«use»

«use»

«use»



«use»

«use»

«use»

«aapFunctionalClust...

Log and Trace

+ IsEnabled()

+ Log(MsgId, Params)

+ LogDebug(): LogStream

+ LogError(): LogStream

+ LogFatal(): LogStream

+ LogInfo(): LogStream

+ LogVerbose(): LogStream

+ LogWarn(): LogStream

+ WithLevel(): LogStream

«aapAPI,aapNativeInterface» Logger

**Figure 9.17: Users of the Log and Trace interfaces**



«aapFunctionalCluster» Update and Configuration

Management daemon-based



«aapFunctionalCluster» Adaptive Intrusion Detection System

Manager

daemon-based



«aapFunctionalCluster»

Platform Health Management daemon-based

|  |  |
| --- | --- |
| ***Interface*** | ***Requiring functional clusters*** |
| [Log and Trace](#_bookmark90)::[Logger](#_bookmark91) | [Adaptive Intrusion Detection System Manager](#_bookmark178) |
| [Communication Management](#_bookmark105) |
| [Cryptography](#_bookmark150) |
| [Diagnostic Management](#_bookmark214) |
| [Execution Management](#_bookmark61) |
| [Identity and Access Management](#_bookmark175) |
| [Network Management](#_bookmark112) |
| [Persistency](#_bookmark124) |
| [Platform Health Management](#_bookmark186) |
| [State Management](#_bookmark76) |
| [Time Synchronization](#_bookmark115) |
| [Update and Configuration Management](#_bookmark200) |

**Table 9.5: Interfaces provided by Log and Trace to other Functional Clusters**

* + - 1. **Required interfaces**



«aapFunctionalCluster» Log and Trace

«use» «use»

«aapInternal» TCP/IP Stack

«aapInternal» Non-volatile Storage

**Figure 9.18: Interfaces required by LogAndTrace**



«use»

«aapRequiredPort»

«aapFunctionalCluster» daemon-based Time Synchronization

+ GetCurrentTime()

+ GetRateDeviation()

+ GetTimeWithStatus(): SynchronizedTimeBaseStatus

+ RegisterStatusChangeNotifier()

+ RegisterSynchronizationStateChangeNotifier()

+ RegisterTimeLeapNotifier()

+ RegisterTimePrecisionMeasurementNotifier()

+ RegisterTimeValidationNotification()

+ UnregisterStatusChangeNotifier()

+ UnregisterSynchronizationStateChangeNotifier()

+ UnregisterTimeLeapNotifier()

+ UnregisterTimePrecisionMeasurementNotifier()

+ UnregisterTimeValidationNotification()

«aapAPI,aapPortInterface»

Time Synchronization::SynchronizedTimeBaseConsumer



Operating System

|  |  |
| --- | --- |
| ***Interface*** | ***Purpose*** |
| Non-volatile Storage | [Log and Trace](#_bookmark90) should use this interface to write log messages to a non-volatile storage, e.g., a file in a filesystem. |
| TCP/IP Stack | [Log and Trace](#_bookmark90) shall use this interface to write log messages to an IP-based network stream. |
| [Registry](#_bookmark210)::[ManifestAccessor](#_bookmark211) | [Log and Trace](#_bookmark90) shall use this interface to read information about modeled messages from the Manifests. |
| [Time Synchronization](#_bookmark115)::[SynchronizedTimeBase](#_bookmark120) [Consumer](#_bookmark120) | [Log and Trace](#_bookmark90) shall use this interface to determine the timestamps that are associated with log messages. |

**Table 9.6: Interfaces required by Log and Trace**

### Core

|  |  |
| --- | --- |
| ***Name:*** | Core |
| ***Short Name:*** | core |
| ***Category:*** | Runtime |
| ***Daemon-based:*** | No |
| ***Responsibilities:*** | [Core](#_bookmark96) provides functionality for initialization and de-initialization of the AUTOSAR Runtime for Adaptive Applications as well as termination of Processes. |

* + - 1. **Defined interfaces**



«aapFunctionalCluster» Core

+ Abort()

+ SetAbortHandler()

+ Deinitialize()

+ Initialize()

«aapAPI,aapNativeInterface» Core

«aapAPI,aapNativeInterf...

Core

**Figure 9.19: Interfaces of Core**

* + - * 1. **Interfaces for initialization and de-initialization**

The AUTOSAR Adaptive Platform for Applications needs to be initialized by an appli- cation before it is used (using [Initialize()](#_bookmark98)) and de-initialized after it is no longer used (using [Deinitialize()](#_bookmark97)).

|  |  |  |
| --- | --- | --- |
| ***Name:*** | InitAndShutdown | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides global initialization and shutdown functions that initialize respectively de-initialize data structures and threads of the AUTOSAR Runtime for Adaptive Applications. | |
| ***Operations:*** | Deinitialize | Destroy all data structures and threads of the AUTOSAR Adaptive Runtime for Applications. After this call, no interaction with the AUTOSAR Adaptive Runtime for Applications is possible. |
| Initialize | Initializes data structures and threads of the AUTOSAR Adaptive Runtime for Applications. Prior to this call, no interaction with the AUTOSAR Adaptive Runtime for Applications is possible. |

* + - * 1. **Interfaces for process termination**

The AUTOSAR Adaptive Platform for Applications provides an explicit abnormal termi- nation facility using [Abort()](#_bookmark99).

|  |  |  |
| --- | --- | --- |
| ***Name:*** | ProcessTermination | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides operation for abnormal termination of processes. | |
| ***Operations:*** | Abort | Abort the current process. This function will never return to its caller. |
| SetAbortHandler | Set a custom global abort handler function and return the previously installed one. |

* + - 1. **Provided interfaces**

[Core](#_bookmark96) currently provides no interfaces to other Functional Clusters.

* + - 1. **Required interfaces**

[Core](#_bookmark96) currently requires no interfaces.

### Operating System Interface

|  |  |
| --- | --- |
| ***Name:*** | Operating System Interface |
| ***Short Name:*** | n/a |
| ***Category:*** | Runtime |
| ***Daemon-based:*** | No |
| ***Responsibilities:*** | The [Operating System Interface](#_bookmark101) provides functionality for implementing multi-threaded real-time embedded applications and corresponds to the POSIX PSE51 profile. That profile provides support to create Threads that may be executed in parallel on modern multi-core processors and to control their properties such as stack memory or their scheduling. In addition, primitives for shared resource access are provided such as Semaphores or memory locking.  Asynchronous (real-time) signals and message passing enable inter-process communication. High resolution timers and clocks are provided to control real-time behavior precisely. Some input/output functions are provided as well but no file system APIs.  POSIX PSE51 and the [Operating System Interface](#_bookmark101) do not provide any means to execute and control Processes. Processes (of the AUTOSAR Adaptive Platform) are entirely controlled by [Execution Management](#_bookmark61) via interfaces that are implementation specific.  Note that a typical AUTOSAR Adaptive Platform stack will not provide an actual implementation of the [Operating System Interface](#_bookmark101) because all functionality is already provided by standard libraries of the programming language (e.g. Standard C++ Library). |

* + - 1. **Defined interfaces**



Adaptive Application

«use»



«aapFunctionalCluster» Operating System Interface

«aapAPI,aapNativeInterface» OperatingSystemInterface

**Figure 9.20: Interfaces defined by Operating System Interface**

|  |  |
| --- | --- |
| ***Name:*** | OperatingSystemInterface |
| ***Technology:*** | Native interface |
| ***Usage:*** | Public API |
| ***Description:*** | This interface represents the POSIX PSE51 profile API. The API is not detailed in this document. |

* + - 1. **Provided interfaces**



«aapFunctionalCluster» Communication Management

«use»



«aapFunctionalCluster» Operating System Interface

«aapAPI,aapNativeInterface» OperatingSystemInterface

**Figure 9.21: Users of the OperatingSystemInterface interfaces**

|  |  |
| --- | --- |
| ***Interface*** | ***Requiring functional clusters*** |
| [Operating System Interface](#_bookmark101)::[OperatingSystem](#_bookmark102) [Interface](#_bookmark102) | [Communication Management](#_bookmark105) |

**Table 9.7: Interfaces provided by Operating System Interface to other Functional Clus- ters**

* + - 1. **Required interfaces**



«aapFunctionalCluster» Operating System Interface

«use»



Operating System

«aapInternal»

Single-Process POSIX API

**Figure 9.22: Interfaces required by Operating System Interface**

|  |  |
| --- | --- |
| ***Interface*** | ***Purpose*** |
| Single-Process POSIX API | [Operating System Interface](#_bookmark101) uses this interface to provide the functionality of the POSIX PSE51 profile, e.g. Threads. Usually the whole POSIX PSE51 profile API will already be provided by the underlying operating system. In this case, the [Operating System](#_bookmark101) [Interface](#_bookmark101) will not have an implementation in the Adaptive Platform stack. |

**Table 9.8: Interfaces required by Operating System Interface**

## Communication



«aapFunctionalCluster» Communication Management



«aapFunctionalCluster» Network Management



«aapFunctionalCluster»

Time Synchronization

daemon-based

**Figure 9.23: Functional Clusters in category Communication**

### Communication Management

|  |  |
| --- | --- |
| ***Name:*** | Communication Management |
| ***Short Name:*** | com |
| ***Category:*** | Communication |
| ***Daemon-based:*** | No |

*Δ*

|  |  |
| --- | --- |
| ***Responsibilities:*** | [Communication Management](#_bookmark105) is responsible for all levels of service-oriented and raw communication between applications in a distributed real-time embedded environment. That is, intra-process communication, inter-process communication and inter-machine communication. The latter is also possible with AUTOSAR Classic Platforms and third-party platforms. Communication paths can be established at design-, start-up-, and run-time. [Communication Management](#_bookmark105) consists of a generic part that handles brokering and configuration as well as generated skeletons for service providers and respective proxies for service consumers. |

* + - 1. **Defined interfaces**

The interfaces of [Communication Management](#_bookmark105) are categorized into interfaces for raw data streams (see Section [9.3.1.1.1](#_bookmark106)), interfaces for SecOC (see Section [9.3.1.1.2](#_bookmark108)), and interfaces freshness value management (see Section [9.3.1.1.3](#_bookmark109)). Please note that a implementation of [Communication Management](#_bookmark105) will generate additional inter- faces for each modeled Service, e.g. a Proxy and a Skeleton interface. However, these generated interfaces are not covered in this document.

* + - * 1. **Interfaces for raw data streams**



Adaptive Application

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | «use» |  |  | «use» |
|  | «aapAPI,aapPortInterface» RawDataStreamClient |  |  | «aapAPI,aapPortInterface» RawDataStreamServer |
| +  +  +  + | Connect() ReadData() Shutdown() WriteData() |  | +  +  +  + | ReadData() Shutdown() WaitForConnection() WriteData() |
|  | «aapProvidedPort» |  |  | «aapProvidedPort» |



«aapFunctionalCluster» Communication Management

**Figure 9.24: Interfaces for raw data streams**

|  |  |
| --- | --- |
| ***Name:*** | RawDataStreamClient |
| ***Technology:*** | Port interface |
| ***Generated:*** | No |
| ***Meta-model interface type:*** | RawDataStreamClientInterface |
| ***Usage:*** | Public API |
| ***Description:*** | This interface provides functionality for a client reading and writing binary data streams over a network connection. |

*Δ*

|  |  |  |
| --- | --- | --- |
| ***Operations:*** | Connect | Sets up a socket connection for the raw data stream defined by the instance, and establish a connection to the TCP server. |
| ReadData | Requests to read a number of bytes of data from the socket connection for the raw data stream defined by the instance. |
| Shutdown | Closes the socket connection for the raw data stream defined by the instance. Both, the receiving and the sending part of the socket connection shall be shut down. |
| WriteData | Requests to write of a number of bytes to the socket connection for the raw data stream defined by the instance. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | RawDataStreamServer | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | RawDataStreamServerInterface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides functionality for a server reading and writing binary data streams over a network connection. | |
| ***Operations:*** | ReadData | Requests to read a number of bytes of data from the socket connection for the raw data stream defined by the instance. |
| Shutdown | Closes the socket connection for the raw data stream defined by the instance. Both the receiving and the sending part of the socket connection shall be shut down. |
| WaitForConnection | Initializes the socket and prepare the [RawDataStreamServer](#_bookmark107) instance for incoming connections. |
| WriteData | Requests to write of a number of bytes to the socket connection for the raw data stream defined by the instance. |

* + - * 1. **Interfaces for SecOC**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adaptive Application | | | | |
| «use»  «aapRequiredPort» |  | «use»  «aapRequiredPort» |  | «use»  «aapRequiredPort» |
| «aapAraComServiceInterface,aapAPI» VerificationStatusConfigurationByDataId |  | «aapAraComServiceInterface,aapAPI» VerificationStatusConfigurationByFreshnessId |  | «aapAraComServiceInterface,aapA...  VerificationStatus |
| «aapAccessControlled, aapServiceMeth...  + VerifyStatusOverride() |  | «aapAccessControlled, aapServiceMethod»  + VerifyStatusOverride() |  | «aapAccessControlled, aapServiceE...  + VerificationStatus |
| «aapProvidedPort» |  | «aapProvidedPort» |  | «aapProvidedPort» |



«aapFunctionalCluster» Communication Management

**Figure 9.25: Interfaces for SecOC**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | VerificationStatus | |
| ***Technology:*** | ara::com service interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides an event to informed about the verification status of messages. | |
| ***Events:*** | VerificationStatus | This event is fired for each verification and holds the verification status. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | VerificationStatusConfigurationByDataId | |
| ***Technology:*** | ara::com service interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides functionality to control the verification status of messages. | |
| ***Operations:*** | VerifyStatusOverride | This service method provides the ability to force to accept or to drop a message with or without performing the verification of authenticator or independent of the authenticator verification result, and to force a specific result allowing additional fault handling in the application. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | VerificationStatusConfigurationByFreshnessId | |
| ***Technology:*** | ara::com service interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides functionality to control the verification status of messages. | |
| ***Operations:*** | VerifyStatusOverride | This service method provides the ability to force to accept or to drop a message with or without performing the verification of authenticator or independent of the authenticator verification result, and to force a specific result allowing additional fault handling in the application. |

* + - * 1. **Interfaces for freshness value management**

+ GetRxFreshness()

+ GetTxFreshness()

+ Initialize()

«aapNativeInterface,aapPlatformExtension» FVM

«use»



«aapFunctionalCluster» Communication Management

**Figure 9.26: Interfaces for freshness value management**

|  |  |
| --- | --- |
| ***Name:*** | FVM |
| ***Technology:*** | Native interface |
| ***Usage:*** | Platform extension |

*Δ*

|  |  |  |
| --- | --- | --- |
| ***Description:*** | This interface provides functionality for freshness value management. | |
| ***Operations:*** | GetRxFreshness | Obtain the current freshness value for received messages. |
| GetTxFreshness | Obtain the current freshness value for transmitted messages. |
| Initialize | Initializes freshness value manager plugin implementation. |

* + - 1. **Provided interfaces**

[Communication Management](#_bookmark105) currently provides no interfaces to other Func- tional Clusters.

* + - 1. **Required interfaces**

«use»



«aapFunctionalCluster» Communication Management

«use»

«use»

«use» «use»



Operating System



«aapFunctionalCluster» Registry



«aapFunctionalClust...

Cryptography daemon-based



«aapFunctionalClust...

Log and Trace

+ IsEnabled()

+ Log(MsgId, Params)

+ LogDebug(): LogStream

+ LogError(): LogStream

+ LogFatal(): LogStream

+ LogInfo(): LogStream

+ LogVerbose(): LogStream

+ LogWarn(): LogStream

+ WithLevel(): LogStream

«aapAPI,aapNativeInterf...

Logger

«aapAPI,aapNativeInterf...

CryptoStack



«aapFunctionalCluster» Identity and Access Management

+ CheckAccess()

«aapInternal,aapNativeInt...

PolicyDecisionPoint

«aapInternal,aapNativeInte...

ManifestAccessor

«aapInternal» TCP/IP Stack

**Figure 9.27: Interfaces required by Communication Management**

|  |  |
| --- | --- |
| ***Interface*** | ***Purpose*** |
| TCP/IP Stack | [Communication Management](#_bookmark105) shall use this interface to establish and control TCP/IP-based network connections. |
| [Adaptive Intrusion Detection System Manager](#_bookmark178):: [EventReporter](#_bookmark179) | [Communication Management](#_bookmark105) shall use this interface to report security events. |
| [Communication Management](#_bookmark105)::[FVM](#_bookmark110) | [Communication Management](#_bookmark105) shall use this interface to get freshness values. |
| [Cryptography](#_bookmark150)::[CryptoStack](#_bookmark151) | [Communication Management](#_bookmark105) shall use this interface to establish encrypted connections and generate / verify checksums (MAC). |
| [Identity and Access Management](#_bookmark175)::[Policy](#_bookmark176) [DecisionPoint](#_bookmark176) | [Communication Management](#_bookmark105) shall use this interface to check access to ara::com service methods, fields, and events. |

*Δ*

|  |  |
| --- | --- |
| ***Interface*** | ***Purpose*** |
| [Log and Trace](#_bookmark90)::[Logger](#_bookmark91) | [Communication Management](#_bookmark105) shall use this interface to log standardized messages. |
| [Operating System Interface](#_bookmark101)::[OperatingSystem](#_bookmark102) [Interface](#_bookmark102) | [Communication Management](#_bookmark105) should use this interface to create and control Threads used by the implementation. |
| [Registry](#_bookmark210)::[ManifestAccessor](#_bookmark211) | [Communication Management](#_bookmark105) shall use this interface to read service information from the Manifests. |

**Table 9.9: Interfaces required by Communication Management**

### Network Management

|  |  |
| --- | --- |
| ***Name:*** | Network Management |
| ***Short Name:*** | nm |
| ***Category:*** | Communication |
| ***Daemon-based:*** | No |
| ***Responsibilities:*** | [Network Management](#_bookmark112) provides functionality to request and query the network states for logical network handles, Such network handles can be mapped to physical or partial networks. |

* + - 1. **Defined interfaces**



«aapFunctionalCluster»

State Management

daemon-based

«use»

«aapRequiredPort»

«aapAccessControlled, aapServiceField»

+ NetworkCurrentState

+ NetworkRequestedState

«aapAraComServiceInterface,aapAPI» NetworkState\_{NetworkHandle}

«aapProvidedPort»



«aapFunctionalCluster» Network Management

**Figure 9.28: Interfaces for time base providers**

|  |  |
| --- | --- |
| ***Name:*** | NetworkState\_{NetworkHandle} |
| ***Technology:*** | ara::com service interface |
| ***Usage:*** | Public API |
| ***Description:*** | This interface provides information about network status per NetworkHandle. This interface is intended to be used by [State Management](#_bookmark76) only. |

*Δ*

|  |  |  |
| --- | --- | --- |
| ***Fields:*** | NetworkCurrentState | Indicates if a PNC / VLAN / Physical Network is currently active or not. |
| NetworkRequestedState | Request a PNC / VLAN / Physical Network to get active or to release it. |

* + - 1. **Provided interfaces**



«aapFunctionalCluster»

State Management

daemon-based

«use»

«aapRequiredPort»

«aapAccessControlled, aapServiceField»

+ NetworkCurrentState

+ NetworkRequestedState

«aapAraComServiceInterface,aapAPI» NetworkState\_{NetworkHandle}

«aapProvidedPort»



«aapFunctionalCluster» Network Management

**Figure 9.29: Users of Network Management interfaces**

|  |  |
| --- | --- |
| ***Interface*** | ***Requiring functional clusters*** |
| [Network Management](#_bookmark112)::[NetworkState\_{Network](#_bookmark113) [Handle}](#_bookmark113) | [State Management](#_bookmark76) |

**Table 9.10: Interfaces provided by Network Management to other Functional Clusters**

* + - 1. **Required interfaces**



«aapFunctionalCluster» Network Management

«use» «use»



«aapFunctionalCluster» Log and Trace

+ IsEnabled()

+ Log(MsgId, Params)

+ LogDebug(): LogStream

+ LogError(): LogStream

+ LogFatal(): LogStream

+ LogInfo(): LogStream

+ LogVerbose(): LogStream

+ LogWarn(): LogStream

+ WithLevel(): LogStream

«aapAPI,aapNativeInterface» Log and Trace::Logger

«aapInternal» TCP/IP Stack

**Figure 9.30: Interfaces required by Network Management**



Operating System

|  |  |
| --- | --- |
| ***Interface*** | ***Purpose*** |
| TCP/IP Stack | [Network Management](#_bookmark112) should use this interface to set and to determine the status of IP-based networks. |
| [Log and Trace](#_bookmark90)::[Logger](#_bookmark91) | [Network Management](#_bookmark112) shall use this interface to log standardized messages. |
| [Registry](#_bookmark210)::[ManifestAccessor](#_bookmark211) | [Network Management](#_bookmark112) shall use this interface to read information about  NmNetworkHandles from the Manifests. |

**Table 9.11: Interfaces required by Network Management**

### Time Synchronization

|  |  |
| --- | --- |
| ***Name:*** | Time Synchronization |
| ***Short Name:*** | tsync |
| ***Category:*** | Communication |
| ***Daemon-based:*** | Yes |
| ***Responsibilities:*** | [Time Synchronization](#_bookmark115) provides synchronized time information in distributed applications. Synchronized time information between different applications and/or Machines is of paramount importance when the correlation of different events across a distributed system is needed, either to be able to track such events in time or to trigger them at an accurate point in time. |

* + - 1. **Defined interfaces**

The interfaces of [Time Synchronization](#_bookmark115) are categorized into interfaces for provid- ing time information (see Section [9.3.3.1.1](#_bookmark116)) and interfaces for consuming time informa- tion (see Section [9.3.3.1.2](#_bookmark119)).

* + - * 1. **Interfaces for time base providers**

[Time Synchronization](#_bookmark115) defines the [SynchronizedTimeBaseProvider](#_bookmark117) and [Offset-](#_bookmark118) [TimeBaseProvider](#_bookmark118) interfaces to provide time information for a synchronized time base.



Adaptive Application



«use»

«aapProvidedPort»

«use»

«aapProvidedPort»

«aapFunctionalCluster» Time Synchronization

daemon-based

+ GetCurrentTime()

+ GetRateCorrection()

+ GetUserData()

+ RegisterTimeValidationNotification()

+ SetOffsetTime()

+ SetRateCorrection()

+ SetUserData()

+ UnregisterTimeValidationNotification()

+ GetCurrentTime()

+ GetRateCorrection()

+ GetUserData()

+ RegisterTimeValidationNotification()

+ SetRateCorrection()

+ SetTime()

+ SetUserData()

+ UnregisterTimeValidationNotification()

+ UpdateTime()

«aapPortInterface,aapAPI» Time Synchronization:: OffsetTimeBaseProvider

«aapAPI,aapPortInterface» Time Synchronization::

SynchronizedTimeBaseProvider

**Figure 9.31: Interfaces for time base providers**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | SynchronizedTimeBaseProvider | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | SynchronizedTimeBaseProviderInterface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Provides access to a synchronized time base. It allows to get the current time point, the rate deviation, the current status and the received user data. | |
| ***Operations:*** | GetCurrentTime | Obtain the current time (regardless of the current sync status). |
| GetRateCorrection | Obtain the current rate deviation of the clock. |
| GetUserData | Get the user defined data of the time base. |
| RegisterTimeValidationNotification | Used by time provider applications to receive time sync parameters. A maximum of one notifier can be registered. Every further registration overwrites the current registration. |
| SetRateCorrection | Set the rate correction that will be applied to time values. |
| SetTime | Set a new time value for the clock. Setting a new time also triggers transmission on the bus. |

*q*

*Δ*

|  |  |  |
| --- | --- | --- |
|  | SetUserData | Set the user data of the time base. |
| UnregisterTimeValidationNotification | Used by time provider applications to receive time sync parameters. |
| UpdateTime | Set a new time value for the clock. The clock value is only updated locally, transmission on the bus will happen in the next cycle. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | OffsetTimeBaseProvider | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | SynchronizedTimeBaseProviderInterface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Provides access to the offset time base. It allows to get the current time point, the rate deviation, the current status and the received user data. | |
| ***Operations:*** | GetCurrentTime | Get the current time (regardless of the current sync status). |
| GetRateCorrection | Obtain the current rate deviation of the clock. |
| GetUserData | Get the user defined data of the time base. |
| RegisterTimeValidationNotification | Used by time provider applications to receive time sync parameters. A maximum of one notifier can be registered. Every further registration overwrites the current registration. |
| SetOffsetTime | Set a new offset time value for the clock. Setting a new time also triggers transmission on the bus. |
| SetRateCorrection | Set the rate correction that will be applied to time values. |
| SetUserData | Set the user data of the time base. |
| UnregisterTimeValidationNotification | Un-register the notifier to receive time sync parameters. |

* + - * 1. **Interfaces for time base consumers**

[Time Synchronization](#_bookmark115) defines the [SynchronizedTimeBaseConsumer](#_bookmark120) interface to retrieve time information for a synchronized time base. [SynchronizedTimeBaseStatus](#_bookmark121) is used to determine the status of a synchronized time base.



«use»

«aapRequiredPort»

«use»

«aapFunctionalCluster» Time Synchronization

daemon-based

+ GetCreationTime()

+ GetLeapJump()

+ GetSynchronizationStatus()

+ GetUserData()

+ GetCurrentTime()

+ GetRateDeviation()

+ GetTimeWithStatus(): SynchronizedTimeBaseStatus

+ RegisterStatusChangeNotifier()

+ RegisterSynchronizationStateChangeNotifier()

+ RegisterTimeLeapNotifier()

+ RegisterTimePrecisionMeasurementNotifier()

+ RegisterTimeValidationNotification()

+ UnregisterStatusChangeNotifier()

+ UnregisterSynchronizationStateChangeNotifier()

+ UnregisterTimeLeapNotifier()

+ UnregisterTimePrecisionMeasurementNotifier()

+ UnregisterTimeValidationNotification()

«aapAPI,aapNativeInterface» Time Synchronization:: SynchronizedTimeBaseStatus

«aapAPI,aapPortInterface» Time Synchronization::

SynchronizedTimeBaseConsumer

Adaptive Application

**Figure 9.32: Interfaces for time base consumers**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | SynchronizedTimeBaseConsumer | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | SynchronizedTimeBaseConsumerInterface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Provides access to the synchronized time base. It allows to get the current time point, the rate deviation, the current status and the received user data. | |
| ***Operations:*** | GetCurrentTime | Obtain the current time (regardless of the current sync status). |
| GetRateDeviation | Obtain the current rate deviation of the clock. |
| GetTimeWithStatus | Obtain a snapshot of the current state of the clock. This includes status flags, clock configuration and the actual time value of the created status object. |
| RegisterStatusChangeNotifier | Register a notifier function which is called if a status flag is changed (i.e. synchronization state, time leap or userdata). A maximum of one notifier can be registered. Every further registration overwrites the current registration. |
| RegisterSynchronizationStateChange Notifier | Register a notifier function which is called if a synchronization state is changed. A maximum of one notifier can be registered. Every further registration overwrites the current registration. |
| RegisterTimeLeapNotifier | Register a notifier function which is called if a time leap happened. A maximum of one notifier can be registered. Every further registration overwrites the current registration. |
| RegisterTimePrecisionMeasurement Notifier | Register a notifier function which is called if a new time precision snapshot is available. A maximum of one notifier can be registered. Every further registration overwrites the current registration. [Time](#_bookmark115) [Synchronization](#_bookmark115) will not do any queuing. If needed it has to be done within the notifier. |

*q*

*Δ*

|  |  |  |
| --- | --- | --- |
|  | RegisterTimeValidationNotification | Used by time consumer applications to receive time sync parameters. A maximum of one notifier can be registered. Every further registration overwrites the current registration. |
| UnregisterStatusChangeNotifier | Un-register a notifier function which is called if a status flag is changed (i.e. synchronization state, time leap or userdata). |
| UnregisterSynchronizationStateChange Notifier | Un-register a notifier function which is called if a synchronization state is changed. |
| UnregisterTimeLeapNotifier | Un-register a notifier function which is called if a time leap happened. |
| UnregisterTimePrecisionMeasurement Notifier | Un-register a notifier function which is called if a new time precision snapshot is available. |
| UnregisterTimeValidationNotification | Un-register a notifier function for receiving time sync parameters. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | SynchronizedTimeBaseStatus | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Represents a snapshot of a time point including its states. | |
| ***Operations:*** | GetCreationTime | Obtain the creation time of this object. |
| GetLeapJump | Determine the direction of a leap jump. Only the jump until the previous object creation is included. |
| GetSynchronizationStatus | Returns the synchronization state when the object was created. |
| GetUserData | Returns the user defined data of the time base. |

* + - 1. **Provided interfaces**



«aapFunctionalCluster» Adaptive Intrusion Detection System

Manager

daemon-based



«aapFunctionalCluster» Update and Configuration Management

daemon-based



«use»

«aapRequiredPort»

«use»

«aapRequiredPort»

«use»

«aapRequiredPort»

«use»

«aapRequiredPort»

«use»

«aapRequiredPort»

daemon-based

«aapFunctionalCluster» Time Synchronization

+ GetCurrentTime()

+ GetRateDeviation()

+ GetTimeWithStatus(): SynchronizedTimeBaseStatus

+ RegisterStatusChangeNotifier()

+ RegisterSynchronizationStateChangeNotifier()

+ RegisterTimeLeapNotifier()

+ RegisterTimePrecisionMeasurementNotifier()

+ RegisterTimeValidationNotification()

+ UnregisterStatusChangeNotifier()

+ UnregisterSynchronizationStateChangeNotifier()

+ UnregisterTimeLeapNotifier()

+ UnregisterTimePrecisionMeasurementNotifier()

+ UnregisterTimeValidationNotification()

«aapAPI,aapPortInterface» SynchronizedTimeBaseConsumer

«aapFunctionalCluster» Persistency

«aapFunctionalCluster»

Execution Management daemon-based

«aapFunctionalCluster» Log and Trace

**Figure 9.33: Users of Time Synchronization interfaces**

|  |  |
| --- | --- |
| ***Interface*** | ***Requiring functional clusters*** |
| [Time Synchronization](#_bookmark115)::[SynchronizedTimeBase](#_bookmark120) [Consumer](#_bookmark120) | [Adaptive Intrusion Detection System Manager](#_bookmark178) |
| [Execution Management](#_bookmark61) |
| [Log and Trace](#_bookmark90) |
| [Persistency](#_bookmark124) |
| [Update and Configuration Management](#_bookmark200) |

**Table 9.12: Interfaces provided by Time Synchronization to other Functional Clusters**

* + - 1. **Required interfaces**

«use»



daemon-based

«aapFunctionalCluster» Time Synchronization

«use»

«use»



«aapInternal» Raw Socket API

Operating System



«aapAPI,aapNativeInterf... Log and Trace::Logger

+ IsEnabled()

+ Log(MsgId, Params)

+ LogDebug(): LogStream

+ LogError(): LogStream

+ LogFatal(): LogStream

+ LogInfo(): LogStream

+ LogVerbose(): LogStream

+ LogWarn(): LogStream

+ WithLevel(): LogStream

«aapFunctionalClust...

Log and Trace



«aapAPI,aapPortInterface» Persistency::KeyValueStorage

+ DiscardPendingChanges()

+ GetAllKeys()

+ GetValue()

+ KeyExists()

+ RecoverKey()

+ RemoveAllKeys()

+ RemoveKey()

+ ResetKey()

+ SetValue()

+ SyncToStorage()

«aapFunctionalClust...

Persistency

**Figure 9.34: Interfaces required by Time Synchronization**

|  |  |
| --- | --- |
| ***Interface*** | ***Purpose*** |
| Raw Socket API | [Time Synchronization](#_bookmark115) should use this interface to send and receive raw ethernet packets as required by the time synchronization protocol. |
| [Execution Management](#_bookmark61)::[ExecutionClient](#_bookmark64) | [Time Synchronization](#_bookmark115) shall use this interface to report the state of its daemon process. |
| [Log and Trace](#_bookmark90)::[Logger](#_bookmark91) | [Time Synchronization](#_bookmark115) shall use this interface to log standardized messages. |
| [Persistency](#_bookmark124)::[KeyValueStorageOperations](#_bookmark138) | [Time Synchronization](#_bookmark115) should use this interface to persist the last received timestamp to enable a faster startup. |
| [Persistency](#_bookmark124)::[KeyValueStorage](#_bookmark141) | [Time Synchronization](#_bookmark115) should use this interface to persist the last received timestamp to enable a faster startup. |
| [Platform Health Management](#_bookmark186)::[SupervisedEntity](#_bookmark189) | [Time Synchronization](#_bookmark115) should use this interface to enable supervision of its daemon process by [Platform Health Management](#_bookmark186) |
| [Registry](#_bookmark210)::[ManifestAccessor](#_bookmark211) | [Time Synchronization](#_bookmark115) shall use this interface to read information about TimeBaseResources as well as their providers and consumers from the Manifests. |

**Table 9.13: Interfaces required by Time Synchronization**

## Storage



«aapFunctionalCluster» Persistency

**Figure 9.35: Functional Clusters in category Storage**

### Persistency

|  |  |
| --- | --- |
| ***Name:*** | Persistency |
| ***Short Name:*** | per |
| ***Category:*** | Storage |
| ***Daemon-based:*** | No |
| ***Responsibilities:*** | [Persistency](#_bookmark124) provides functionality to store and retrieve information to/from non-volatile storage of a Machine.  Persistent data is always private to one Process and is persisted across boot and ignition cycles. There is no mechanism available to share data between different Processes using [Persistency](#_bookmark124) to prevent a second path of data exchange besides [Communication Management](#_bookmark105). However, [Persistency](#_bookmark124) supports concurrent access from multiple threads of the same application running in the context of the same Process.  [Persistency](#_bookmark124) offers integrity of the stored data and provides error detection and correction schemes. [Persistency](#_bookmark124) also offers confidentiality of the stored data using encryption.  [Persistency](#_bookmark124) provides statistics, for example, the used storage space. |

* + - 1. **Defined interfaces**

The interfaces of [Persistency](#_bookmark124) are categorized into interfaces for file access (see Section [9.4.1.1.1](#_bookmark125)), interfaces for a key-value-based data access (see Section [9.4.1.1.2](#_bookmark136)) and interfaces for general management of persistent data (see Section [9.4.1.1.3](#_bookmark145)).

* + - * 1. **Interfaces for file storage**

[Persistency](#_bookmark124) provides read and write access to plain files by means of a [FileStor-](#_bookmark130) [age](#_bookmark130) (cf. Figure [9.36](#_bookmark126)). A [FileStorage](#_bookmark130) has to be opened using [OpenFileStorage()](#_bookmark129). A [FileStorage](#_bookmark130) then provides access to several files using their name.



Adaptive Application

«use»

«use»

«use»

«use»

«aapAPI,aapNativeInterface» Persistency



«aapAPI,aapPortInterface» Persistency::FileStorage

«aapAPI,aapNativeInterface» Persistency::ReadAccessor

«aapAPI,aapNativeInterface» Persistency::ReadWriteAccessor

+ GetCurrentFileStorageSize()

+ OpenFileStorage(): FileStorage

+ RecoverAllFiles()

+ ResetAllFiles()

+ DeleteFile()

+ FileExists()

+ GetAllFileNames()

+ GetCurrentFileSize()

+ GetFileInfo()

+ OpenFileReadOnly(): ReadAccessor

+ OpenFileReadWrite(): ReadWriteAccessor

+ OpenFileWriteOnly(): ReadWriteAccessor

+ RecoverFile()

+ ResetFile()

+ GetByte()

+ GetChar()

+ GetPosition()

+ GetSize()

+ IsEof()

+ MovePosition()

+ PeekByte()

+ PeekChar()

+ ReadBinary()

+ ReadLine()

+ ReadText()

+ SetPosition()

+ operator<<()

+ SetFileSize()

+ SyncToFile()

+ WriteBinary()

+ WriteText()

«aapFunctionalCluster» Persistency

**Figure 9.36:** **Interfaces for file storage**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | FileStorageOperations | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides functions to open and manage [FileStorage](#_bookmark130)s. | |
| ***Operations:*** | GetCurrentFileStorageSize | Returns the space in bytes currently occupied by a [FileStorage](#_bookmark130). |
| OpenFileStorage | Opens a [FileStorage](#_bookmark130). |
| RecoverAllFiles | Recovers a [FileStorage](#_bookmark130) including all files. |
| ResetAllFiles | Resets a [FileStorage](#_bookmark130) including all files. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | FileStorage | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | PersistencyFileStorageInterface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides functions to open and manage files. | |
| ***Operations:*** | DeleteFile | Deletes a file from this [FileStorage](#_bookmark130). |
| FileExists | Checks if a file exists in this [FileStorage](#_bookmark130). |
| GetAllFileNames | Returns a list of all currently available files of this [FileStorage](#_bookmark130). |
| GetCurrentFileSize | Returns the space in bytes currently occupied by the content of a file of this [FileStorage](#_bookmark130). |
| GetFileInfo | Returns additional information on a file of this [FileStorage](#_bookmark130). |
| OpenFileReadOnly | Opens a file of this [FileStorage](#_bookmark130) for reading. |
| OpenFileReadWrite | Opens a file of this [FileStorage](#_bookmark130) for reading and writing. |
| OpenFileWriteOnly | Opens a file of this [FileStorage](#_bookmark130) for writing. |

*q*

*Δ*

|  |  |  |
| --- | --- | --- |
|  | RecoverFile | Recovers a file of this [FileStorage](#_bookmark130). |
| ResetFile | Resets a file of this [FileStorage](#_bookmark130) to its initial content. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | ReadAccessor | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides functions to read text and binary data from a file. | |
| ***Operations:*** | GetByte | Returns the byte at the current position of the file, advancing the current position. |
| GetChar | Returns the character at the current position of the file, advancing the current position. |
| GetPosition | Returns the current position relative to the beginning of the file. |
| GetSize | Returns the current size of a file in bytes. |
| IsEof | Checks if the current position is at end of file. |
| MovePosition | Moves the current position in the file relative to the origin. |
| PeekByte | Returns the byte at the current position of the file. |
| PeekChar | Returns the character at the current position of the file. |
| ReadBinary | Reads all remaining bytes into a Vector of Byte, starting from the current position. |
| ReadLine | Reads a complete line of characters into a String, advancing the current position accordingly. |
| ReadText | Reads all remaining characters into a String, starting from the current position. |
| SetPosition | Sets the current position relative to the beginning of the file. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | ReadWriteAccessor | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides functions to read and write text and binary data from / to a file. | |
| ***Operations:*** | SetFileSize | Reduces the size of the file to ’size’, effectively removing the current content of the file beyond this size. |
| SyncToFile | Triggers flushing of the current file content to the physical storage. |
| WriteBinary | Writes binary data to the file. |
| WriteText | Writes a string to the file. |
| operator<< | Writes a String to the file. |

* + - * 1. **Interfaces for key-value storage**

[Persistency](#_bookmark124) provides read and write access to data structured as key-value pairs by means of the [KeyValueStorage](#_bookmark141) API (cf. Figure [9.37](#_bookmark137)). A [KeyValueStorage](#_bookmark141) has to be created by calling [OpenKeyValueStorage()](#_bookmark140). A [KeyValueStorage](#_bookmark141) then provides access to data stored for individual keys using the [GetValue()](#_bookmark142) and [SetValue()](#_bookmark143) operations.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adaptive Application | | | | |
|  | «use» |  |  | «use» |
|  | «aapAPI,aapNativeInterface» |  |  | «aapAPI,aapPortInterface» |
|  | Persistency |  |  | Persistency::KeyValueStorage |
| + | GetCurrentKeyValueStorageSize() |  | + | DiscardPendingChanges() |
| + | OpenKeyValueStorage(): KeyValueStorage |  | + | GetAllKeys() |
| + | RecoverKeyValueStorage() |  | + | GetValue() |
| + | ResetKeyValueStorage() |  | + | KeyExists() |
|  |  |  | + | RecoverKey() |
|  |  |  | + | RemoveAllKeys() |
|  |  |  | + | RemoveKey() |
|  |  |  | + | ResetKey() |
|  |  |  | + | SetValue() |
|  |  |  | + | SyncToStorage() |

**Figure 9.37:** **Interfaces for key-value-based data storage**



«aapFunctionalCluster» Persistency

|  |  |  |
| --- | --- | --- |
| ***Name:*** | KeyValueStorageOperations | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides functions to open and manage [KeyValueStorage](#_bookmark141)s. | |
| ***Operations:*** | GetCurrentKeyValueStorageSize | Returns the space in bytes currently occupied by a [KeyValueStorage](#_bookmark141). |
| OpenKeyValueStorage | Opens a [KeyValueStorage](#_bookmark141). |
| RecoverKeyValueStorage | Recovers a [KeyValueStorage](#_bookmark141). |
| ResetKeyValueStorage | Resets a [KeyValueStorage](#_bookmark141) to the initial state. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | KeyValueStorage | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | PersistencyKeyValueStorageInterface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides functions to access values associated with keys. | |
| ***Operations:*** | DiscardPendingChanges | Discards changed key-value pairs of the [KeyValueStorage](#_bookmark141) an re-reads them from the physical storage. |
| GetAllKeys | Returns a list of all currently available keys of this [KeyValueStorage](#_bookmark141). |
| GetValue | Returns the value assigned to a key of this [KeyValueStorage](#_bookmark141). |
| KeyExists | Checks if a key exists in this [KeyValueStorage](#_bookmark141). |
| RecoverKey | Recovers a single key of this [KeyValueStorage](#_bookmark141). |
| RemoveAllKeys | Removes all keys and associated values from this [KeyValueStorage](#_bookmark141). |

*q*

*Δ*

|  |  |  |
| --- | --- | --- |
|  | RemoveKey | Removes a key and the associated value from this [KeyValueStorage](#_bookmark141). |
| ResetKey | Resets a key of this [KeyValueStorage](#_bookmark141) to its initial value. |
| SetValue | Stores a key in this [KeyValueStorage](#_bookmark141). |
| SyncToStorage | Triggers flushing of changed key-value pairs of the [KeyValueStorage](#_bookmark141) to the physical storage. |

* + - * 1. **Interfaces for general persistency handling**

[Persistency](#_bookmark124) provides operations for handling and recovery of persistent data of a

Process (cf. Figure [9.38](#_bookmark146)).



Adaptive Application

«use»



«aapFunctionalCluster» Persistency

+ RegisterApplicationDataUpdateCallback()

+ RegisterRecoveryReportCallback()

+ ResetPersistency()

+ UpdatePersistency()

«aapAPI,aapNativeInterface» Persistency

**Figure 9.38:** **Interfaces for general persistency handling**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | PersistencyHandlingOperations | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides operations manage persistent data. | |
| ***Operations:*** | RegisterApplicationDataUpdate Callback | Registers an application data update callback with  [Persistency](#_bookmark124). |
| RegisterRecoveryReportCallback | Register a recovery reporting callback with  [Persistency](#_bookmark124). |
| ResetPersistency | Resets all [FileStorage](#_bookmark130)s and [KeyValueStorage](#_bookmark141)s by entirely removing their content. |
| UpdatePersistency | Updates all [FileStorage](#_bookmark130)s and [KeyValueStorage](#_bookmark141)s after a new manifest was installed. |

* + - 1. **Provided interfaces**



«aapFunctionalClust... Diagnostic Management

daemon-based



«aapFunctionalCluster» Update and Configuration Management

daemon-based



«use»

«aapRequiredPort»

«use»

«aapRequiredPort»

«aapFunctionalCluster» Persistency

+ DeleteFile()

+ FileExists()

+ GetAllFileNames()

+ GetCurrentFileSize()

+ GetFileInfo()

+ OpenFileReadOnly(): ReadAccessor

+ OpenFileReadWrite(): ReadWriteAccessor

+ OpenFileWriteOnly(): ReadWriteAccessor

+ RecoverFile()

+ ResetFile()

«aapAPI,aapPortInterface» Persistency::FileStorage

**Figure 9.39: Users of the FileStorage interfaces**



«aapFunctionalCluster»

Diagnostic Management

daemon-based



«aapFunctionalCluster» Update and Configuration Management

daemon-based



«use»

«aapRequiredPort»

«use»

«use»

«use»

«aapRequiredPort»

«aapFunctionalCluster» Persistency

+ DiscardPendingChanges()

+ GetAllKeys()

+ GetValue()

+ KeyExists()

+ RecoverKey()

+ RemoveAllKeys()

+ RemoveKey()

+ ResetKey()

+ SetValue()

+ SyncToStorage()

«aapAPI,aapPortInterface» Persistency::KeyValueStorage

«aapFunctionalCluster»

Time Synchronization

daemon-based

«aapFunctionalCluster»

State Management

daemon-based

**Figure 9.40: Users of the KeyValueStorage interfaces**

|  |  |
| --- | --- |
| ***Interface*** | ***Requiring functional clusters*** |
| [Persistency](#_bookmark124)::[PersistencyHandlingOperations](#_bookmark147) | [Diagnostic Management](#_bookmark214) |
| [Update and Configuration Management](#_bookmark200) |
| [Persistency](#_bookmark124)::[FileStorageOperations](#_bookmark127) | [Diagnostic Management](#_bookmark214) |
| [Update and Configuration Management](#_bookmark200) |
| [Persistency](#_bookmark124)::[KeyValueStorageOperations](#_bookmark138) | [Diagnostic Management](#_bookmark214) |
| [State Management](#_bookmark76) |
| [Time Synchronization](#_bookmark115) |
| [Update and Configuration Management](#_bookmark200) |

*Δ*

|  |  |
| --- | --- |
| ***Interface*** | ***Requiring functional clusters*** |
| [Persistency](#_bookmark124)::[FileStorage](#_bookmark130) | [Diagnostic Management](#_bookmark214) |
| [Update and Configuration Management](#_bookmark200) |
| [Persistency](#_bookmark124)::[KeyValueStorage](#_bookmark141) | [Diagnostic Management](#_bookmark214) |
| [Execution Management](#_bookmark61) |
| [State Management](#_bookmark76) |
| [Time Synchronization](#_bookmark115) |
| [Update and Configuration Management](#_bookmark200) |
| [Persistency](#_bookmark124)::[ReadAccessor](#_bookmark134) | [Diagnostic Management](#_bookmark214) |
| [Update and Configuration Management](#_bookmark200) |
| [Persistency](#_bookmark124)::[ReadWriteAccessor](#_bookmark135) | [Diagnostic Management](#_bookmark214) |
| [Update and Configuration Management](#_bookmark200) |

**Table 9.14: Interfaces provided by Persistency to other Functional Clusters**

* + - 1. **Required interfaces**



«aapFunctionalCluster» Persistency

«use»

«use» «use»

«aapAPI,aapNativeInterf... Cryptography::CryptoStack



«aapFunctionalClust...

Log and Trace

+ IsEnabled()

+ Log(MsgId, Params)

+ LogDebug(): LogStream

+ LogError(): LogStream

+ LogFatal(): LogStream

+ LogInfo(): LogStream

+ LogVerbose(): LogStream

+ LogWarn(): LogStream

+ WithLevel(): LogStream

«aapAPI,aapNativeInterf... Log and Trace::Logger

**Figure 9.41: Interfaces required by Persistency**



«aapFunctionalClust...

Cryptography daemon-based



Operating System



«use»

«aapRequiredPort»

daemon-based

«aapFunctionalCluster» Time Synchronization

+ GetCurrentTime()

+ GetRateDeviation()

+ GetTimeWithStatus(): SynchronizedTimeBaseStatus

+ RegisterStatusChangeNotifier()

+ RegisterSynchronizationStateChangeNotifier()

+ RegisterTimeLeapNotifier()

+ RegisterTimePrecisionMeasurementNotifier()

+ RegisterTimeValidationNotification()

+ UnregisterStatusChangeNotifier()

+ UnregisterSynchronizationStateChangeNotifier()

+ UnregisterTimeLeapNotifier()

+ UnregisterTimePrecisionMeasurementNotifier()

+ UnregisterTimeValidationNotification()

«aapAPI,aapPortInterface»

Time Synchronization::SynchronizedTimeBaseConsumer

«aapInternal» Non-volatile Storage

|  |  |
| --- | --- |
| ***Interface*** | ***Purpose*** |
| Non-volatile Storage | [Persistency](#_bookmark124) uses this interface to access the non-volatile storage provided by the underlying operating system, for example, a file system. |
| [Cryptography](#_bookmark150)::[CryptoStack](#_bookmark151) | [Persistency](#_bookmark124) uses this interface to ensure confidentiality and integrity of the persisted data. |
| [Log and Trace](#_bookmark90)::[Logger](#_bookmark91) | [Persistency](#_bookmark124) shall use this interface to log standardized messages. |

*Δ*

|  |  |
| --- | --- |
| ***Interface*** | ***Purpose*** |
| [Registry](#_bookmark210)::[ManifestAccessor](#_bookmark211) | [Persistency](#_bookmark124) shall use this interface to read its configuration information from the Manifests. |
| [Time Synchronization](#_bookmark115)::[SynchronizedTimeBase](#_bookmark120) [Consumer](#_bookmark120) | [Persistency](#_bookmark124) should use this interface to determine timestamps included in the meta-information of files, e.g., modification timestamp. |

**Table 9.15: Interfaces required by Persistency**

## Security



«aapFunctionalCluster»

Cryptography

daemon-based



«aapFunctionalCluster» Identity and Access Management



«aapFunctionalCluster» Adaptive Intrusion Detection System Manager

daemon-based



«aapFunctionalCl...

Firewall

daemon-based

**Figure 9.42: Functional Clusters in category Security**

### Cryptography

|  |  |
| --- | --- |
| ***Name:*** | Cryptography |
| ***Short Name:*** | crypto |
| ***Category:*** | Security |
| ***Daemon-based:*** | Yes |
| ***Responsibilities:*** | [Cryptography](#_bookmark150) provides various cryptographic routines to ensure confidentiality of data, to ensure integrity of data (e.g., using hashes), and auxiliary functions for example key management and random number generation. [Cryptography](#_bookmark150) is designed to support encapsulation of  security-sensitive operations and decisions in a separate component, such as a Hardware Security Module (HSM). Additional protection of keys and key usage can be provided by constraining keys to particular usages (e.g., decrypt-only), or limiting the availability of keys to individual applications as reported by [Identity and Access Management](#_bookmark175).  Depending on application support, [Cryptography](#_bookmark150) can also be used to protect session keys and intermediate secrets when processing cryptographic protocols such as TLS and SecOC. |

* + - 1. **Defined interfaces**
         1. **Common interfaces**

The main entry point for using the [Cryptography](#_bookmark150) API are the factory functions [Load-](#_bookmark152) [CryptoProvider()](#_bookmark152) for using cryptographic routines, [LoadKeyStorageProvider()](#_bookmark153) for ac- cess to the key store, and [LoadX509Provider()](#_bookmark154) for X.509 certificate handling.



Adaptive Application

«use»



«use» «use»

«use»

«aapAPI,aapNativeInterface»

+ GenerateRandomData()

+ GetSecureCounter()

+ LoadCryptoProvider()

+ LoadKeyStorageProvider()

+ LoadX509Provider()

«aapAPI,aapNativeInt...

IOInterface

+ GetAllowedUsage()

+ GetCapacity()

+ GetCryptoObjectType()

+ GetObjectId()

+ GetPayloadSize()

+ GetPrimitiveId()

+ GetTypeRestriction()

+ IsObjectExportable()

+ IsObjectSession()

+ IsValid()

+ IsVolatile()

+ IsWritable()

«aapAPI,aapNativeInt...

Serializable

+ ExportPublicly()

«aapAPI,aapNativeInt... VolatileTrustedContainer

+ GetIOInterface()

daemon-based

«aapFunctionalCluster» Cryptography

**Figure 9.43: Common Interfaces of Cryptography**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | EntryPoint | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides the main entry points for using the [Cryptography](#_bookmark150) API. | |
| ***Operations:*** | GenerateRandomData | Generate random data of the requested size. |
| GetSecureCounter | Get current value of 128 bit SecureCounter supported by the Crypto Stack. |
| LoadCryptoProvider | Factory that creates or returns existing single instance of a specific [CryptoProvider](#_bookmark157). |
| LoadKeyStorageProvider | Factory that creates or returns an existing single instance of the [KeyStorageProvider](#_bookmark169). |
| LoadX509Provider | Factory that creates or returns an existing single instance of the [X509Provider](#_bookmark172). |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | IOInterface | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Interface for saving and loading of security objects. | |
| ***Operations:*** | GetAllowedUsage | Return actual allowed key/seed usage flags defined by the key slot prototype for this "Actor" and current content of the container. |
| GetCapacity | Return capacity of the underlying resource. |
| GetCryptoObjectType | Return the type of the object referenced by this [IOInterface](#_bookmark155). |
| GetObjectId | Return an ID of an object stored to this [IOInterface](#_bookmark155). |
| GetPayloadSize | Return size of an object payload stored in the underlying buffer. |
| GetPrimitiveId | Get vendor specific ID of the primitive. |
| GetTypeRestriction | Return content type restriction. |

*Δ*

|  |  |  |
| --- | --- | --- |
|  | IsObjectExportable | Return the exportable attribute of an object stored to the container. |
| IsObjectSession | Return the session (or temporary) attribute of an object as set. |
| IsValid | Get whether the underlying [KeySlot](#_bookmark170) is valid. |
| IsVolatile | Return volatility of the underlying buffer. |
| IsWritable | Get whether the underlying [KeySlot](#_bookmark170) is writable. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | Serializable | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Serializable object interface. | |
| ***Operations:*** | ExportPublicly | Serialize itself publicly. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | VolatileTrustedContainer | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface is used for buffering [Cryptography](#_bookmark150) API objects in RAM. | |
| ***Operations:*** | GetIOInterface | Retrieve the [IOInterface](#_bookmark155) used for importing/exporting objects into this container. |

* + - * 1. **General cryptography interfaces**

The [CryptoProvider](#_bookmark157) interface provides access to various cryptographic routines. Each of those routines is managed by specializations of the [CryptoContext](#_bookmark158) interface.



Adaptive Application

«use» «use»



daemon-based

«aapFunctionalCluster» Cryptography

+ GetCryptoPrimitiveId()

+ IsInitialized()

+ MyProvider()

+ AllocVolatileContainer()

+ ConvertToAlgId()

+ ConvertToAlgName()

+ CreateAuthCipherCtx()

+ CreateDecryptorPrivateCtx()

+ CreateEncryptorPublicCtx()

+ CreateHashDigest()

+ CreateHashFunctionCtx()

+ CreateKeyAgreementPrivateCtx()

+ CreateKeyDecapsulatorPrivateCtx()

+ CreateKeyDerivationFunctionCtx()

+ CreateKeyEncapsulatorPublicCtx()

+ CreateMessageAuthCodeCtx()

+ CreateMsgRecoveryPublicCtx()

+ CreateRandomGeneratorCtx()

+ CreateSigEncodePrivateCtx()

+ CreateSignature()

+ CreateSignerPrivateCtx()

+ CreateStreamCipherCtx()

+ CreateSymmetricBlockCipherCtx()

+ CreateSymmetricKeyWrapperCtx()

+ CreateVerifierPublicCtx()

+ ExportPublicObject()

+ ExportSecuredObject()

+ GeneratePrivateKey()

+ GenerateSeed()

+ GenerateSymmetricKey()

+ GetPayloadStorageSize()

+ GetSerializedSize()

+ ImportPublicObject()

+ ImportSecuredObject()

+ LoadObject()

+ LoadPrivateKey()

+ LoadPublicKey()

+ LoadSecretSeed()

+ LoadSymmetricKey()

«aapAPI,aapNativeInterface» CryptoContext

«aapAPI,aapNativeInterface» CryptoProvider

**Figure 9.44: General Cryptography Interfaces**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | CryptoProvider | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This is a "factory" interface of all supported crypto primitives and a "trusted environment" for internal communications between them. | |
| ***Operations:*** | AllocVolatileContainer | Allocate a [VolatileTrustedContainer](#_bookmark156) according to directly specified capacity. |
| ConvertToAlgId | Convert a common name of crypto algorithm to a correspondent vendor specific binary algorithm ID. |
| ConvertToAlgName | Convert a vendor specific binary algorithm ID to a correspondent common name of the crypto algorithm. |
| CreateAuthCipherCtx | Create a symmetric authenticated cipher context. |
| CreateDecryptorPrivateCtx | Create a decryption private key context. |
| CreateEncryptorPublicCtx | Create an encryption public key context. |
| CreateHashDigest | Construct signature object from directly provided components of a hash digest. |
| CreateHashFunctionCtx | Create a hash function context. |

*Δ*

|  |  |  |
| --- | --- | --- |
|  | CreateKeyAgreementPrivateCtx | Create a key-agreement private key context. |
| CreateKeyDecapsulatorPrivateCtx | Create a key-decapsulator private key context of a key encapsulation mechanism. |
| CreateKeyDerivationFunctionCtx | Create a key derivation function context. |
| CreateKeyEncapsulatorPublicCtx | Create a key-encapsulator public key context of a key encapsulation mechanism. |
| CreateMessageAuthCodeCtx | Create a symmetric message authentication code context. |
| CreateMsgRecoveryPublicCtx | Create a message recovery public key context. |
| CreateRandomGeneratorCtx | Create a random number generator context. |
| CreateSigEncodePrivateCtx | Create a signature encoding private key context. |
| CreateSignature | Construct a signature object from directly provided components of a digital signature/MAC or authenticated encryption (AE/AEAD). |
| CreateSignerPrivateCtx | Create a signature private key context. |
| CreateStreamCipherCtx | Create a symmetric stream cipher context. |
| CreateSymmetricBlockCipherCtx | Create a symmetric block cipher context. |
| CreateSymmetricKeyWrapperCtx | Create a symmetric key-wrap algorithm context. |
| CreateVerifierPublicCtx | Create a signature verification public key context. |
| ExportPublicObject | Export publicly an object. |
| ExportSecuredObject | Export a crypto object in a secure manner. |
| GeneratePrivateKey | Allocate a new private key context of correspondent type and generates the key value randomly. |
| GenerateSeed | Generate a random Secret Seed object of requested algorithm. |
| GenerateSymmetricKey | Allocate a new symmetric key object and fill it by a new randomly generated value. |
| GetPayloadStorageSize | Return minimally required capacity of a key slot for saving of the object’s payload. |
| GetSerializedSize | Return required buffer size for serialization of an object in specific format. |
| ImportPublicObject | Import publicly serialized object to a storage location. |
| ImportSecuredObject | Import securely serialized object to the persistent or volatile storage. |
| LoadObject | Load any crypto object from the [IOInterface](#_bookmark155) provided. |
| LoadPrivateKey | Load a [PrivateKey](#_bookmark168) from the [IOInterface](#_bookmark155) provided. |
| LoadPublicKey | Load a [PublicKey](#_bookmark167) from the [IOInterface](#_bookmark155) provided. |
| LoadSecretSeed | Load a [SecretSeed](#_bookmark165) from the [IOInterface](#_bookmark155) provided. |
| LoadSymmetricKey | Load a [SymmetricKey](#_bookmark166) from the [IOInterface](#_bookmark155) provided. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | CryptoContext | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | A common interface of a mutable cryptographic context, i.e. that is not bound to a single crypto object. | |
| ***Operations:*** | GetCryptoPrimitiveId | Returns a [CryptoPrimitiveId](#_bookmark164) instance containing instance identification. |

*Δ*

|  |  |  |
| --- | --- | --- |
|  | IsInitialized | Check if the crypto context is already initialized and ready to use. |
| MyProvider | Get a reference to the [CryptoProvider](#_bookmark157) of this context. |

* + - * 1. **Cryptography context interfaces**



Adaptive Application

«use» «use»



daemon-based

«aapFunctionalCluster» Cryptography

+ GetActualIvBitLength()

+ GetBlockSize()

+ GetIvSize()

+ IsValidIvSize()

+ CountBytesInCache()

+ EstimateMaxInputSize()

+ EstimateRequiredCapacity()

+ FinishBytes()

+ GetBlockService()

+ GetTransformation()

+ IsBytewiseMode()

+ IsSeekableMode()

+ ProcessBlocks()

+ ProcessBytes()

+ Reset()

+ Seek()

+ SetKey()

+ Start()

*ExtensionService*

«aapAPI,aapNativeInterface» BlockService

*CryptoContext*

«aapAPI,aapNativeInterface» StreamCipherCtx

**Figure 9.45: BlockService and CryptoContext Interfaces**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | BlockService | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Extension meta-information service for block cipher contexts. | |
| ***Operations:*** | GetActualIvBitLength | Get the actual bit-length of an initialization vector loaded to the context. |
| GetBlockSize | Get the block (or internal buffer) size of the base algorithm. |
| GetIvSize | Get default expected size of the initialization vector or nonce. |
| IsValidIvSize | Verify validity of specific initialization vector length. |

|  |  |
| --- | --- |
| ***Name:*** | StreamCipherCtx |
| ***Technology:*** | Native interface |
| ***Usage:*** | Public API |
| ***Description:*** | Generalized stream cipher context interface covering all modes of operation. |

*Δ*

|  |  |  |
| --- | --- | --- |
| ***Operations:*** | CountBytesInCache | Count number of bytes now kept in the context cache. |
| EstimateMaxInputSize | Estimate maximal number of input bytes that may be processed for filling of an output buffer without overflow. |
| EstimateRequiredCapacity | Estimate minimal required capacity of the output buffer, which is enough for saving a result of input data processing. |
| FinishBytes | Process the final part of message (that may be not aligned to the block-size boundary). |
| GetBlockService | Get the [BlockService](#_bookmark159) instance. |
| GetTransformation | Get the kind of transformation configured for this context: Encrypt or Decrypt. |
| IsBytewiseMode | Check the operation mode for the byte-wise property. |
| IsSeekableMode | Check if the seek operation is supported in the current mode. |
| ProcessBlocks | Process initial parts of message aligned to the block-size boundary. |
| ProcessBytes | Process a non-final part of message (that is not aligned to the block-size boundary). |
| Reset | Clear the crypto context. |
| Seek | Set the position of the next byte within the stream of the encryption/decryption gamma. |
| SetKey | Set (deploy) a key to the stream cihper algorithm context. |
| Start | Initialize the context for a new data stream processing or generation (depending from the primitive). |



Adaptive Application

«use» «use» «use» «use»

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | |  | | |  | | |  |
| *CryptoContext*  «aapAPI,aapNativeInterface» EncryptorPrivateCtx | |  | *CryptoContext*  «aapAPI,aapNativeInterface» DecryptorPrivateCtx | |  | *CryptoContext*  «aapAPI,aapNativeInterface» SymmetricBlockCipherCtx | |  | *ExtensionService*  «aapAPI,aapNativeInterface» CryptoService | |
| + GetCryptoService()  + ProcessBlock()  + Reset()  + SetKey() | | + GetCryptoService()  + ProcessBlock()  + Reset()  + SetKey() | | + GetCryptoService()  + GetTransformation()  + ProcessBlock()  + ProcessBlocks()  + Reset()  + SetKey() | | + GetBlockSize()  + GetMaxInputSize()  + GetMaxOutputSize() | |
|  |  | | |  | | |  | | |  |
| «aapFunctionalCluster»  Cryptography  daemon-based | | | | | | | | | | |

**Figure 9.46: CryptoService and CryptoContext Interfaces**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | CryptoService | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Extension meta-information service for cryptographic contexts. | |
| ***Operations:*** | GetBlockSize | Get block (or internal buffer) size of the base algorithm. |

*Δ*

|  |  |  |
| --- | --- | --- |
|  | GetMaxInputSize | Get maximum expected size of the input data block. |
| GetMaxOutputSize | Get maximum possible size of the output data block. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | EncryptorPrivateCtx | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Asymmetric decryption private key context interface. | |
| ***Operations:*** | GetCryptoService | Get the [CryptoService](#_bookmark160) instance. |
| ProcessBlock | Encrypt an input block according to the encryptor configuration. |
| Reset | Clear the crypto context. |
| SetKey | Set (deploy) a key to the decryptor private algorithm context. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | DecryptorPrivateCtx | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Asymmetric decryption private key context interface. | |
| ***Operations:*** | GetCryptoService | Get the [CryptoService](#_bookmark160) instance. |
| ProcessBlock | Decrypt an input block according to the decryptor configuration. |
| Reset | Clear the crypto context. |
| SetKey | Set (deploy) a key to the decryptor private algorithm context. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | SymmetricBlockCipherCtx | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Interface of a symmetric block cipher context with padding. | |
| ***Operations:*** | GetCryptoService | Get the [CryptoService](#_bookmark160) instance. |
| GetTransformation | Get the kind of transformation configured for this context: Encrypt or Decrypt. |
| ProcessBlock | Process (encrypt / decrypt) an input block according to the configuration. |
| ProcessBlocks | Process (encrypt / decrypt) input blocks according to the configuration. |
| Reset | Clear the crypto context. |
| SetKey | Set (deploy) a key to the symmetric algorithm context. |



Adaptive Application

«use»

«use»

«use»

«use»



*CryptoContext CryptoContext CryptoContext BlockService*

«aapAPI,aapNativeInterface» «aapAPI,aapNativeInterface» «aapAPI,aapNativeInterface» «aapAPI,aapNativeInterface» AuthCipherCtx HashFunctionCtx MessageAuthnCodeCtx DigestService

+ Check()

+ GetDigest()

+ GetDigestService()

+ GetMaxAssociatedDataSize()

+ GetTransformation()

+ ProcessConfidentialData()

+ Reset()

+ SetKey()

+ Start()

+ UpdateAssociatedData()

+ Finish()

+ GetDigest()

+ GetDigestService()

+ Start()

+ Update()

+ Check()

+ Finish()

+ GetDigest()

+ GetDigestService()

+ Reset()

+ SetKey()

+ Start()

+ Update()

+ Compare()

+ GetDigestSize()

+ IsFinished()

+ IsStarted()

«aapFunctionalCluster» Cryptography

daemon-based

**Figure 9.47: DigestService and CryptoContext Interfaces**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | DigestService | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Extension meta-information service for digest producing contexts. | |
| ***Operations:*** | Compare | Compare the calculated digest against an expected value. |
| GetDigestSize | Get the output digest size. |
| IsFinished | Check current status of the stream processing: finished or not. |
| IsStarted | Check current status of the stream processing: started or not. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | AuthCipherCtx | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Generalized authenticated cipher context interface. | |
| ***Operations:*** | Check | Check the calculated digest against an expected signature object. |
| GetDigest | Retrieve the calculated digest. |
| GetDigestService | Get the [DigestService](#_bookmark161) instance. |
| GetMaxAssociatedDataSize | Get maximal supported size of associated public data. |
| GetTransformation | Get the kind of transformation configured for this context: Encrypt or Decrypt. |
| ProcessConfidentialData | Process confidential data and return the result. |
| Reset | Clear the crypto context. |
| SetKey | Set (deploy) a key to the authenticated cipher symmetric algorithm context. |
| Start | Initialize the context for a new data processing or generation (depending from the primitive). |

*Δ*

|  |  |  |
| --- | --- | --- |
|  | UpdateAssociatedData | Update the digest calculation by the specified data. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | HashFunctionCtx | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Hash function interface. | |
| ***Operations:*** | Finish | Finish the digest calculation and optionally produce the "signature" object. |
| GetDigest | Get requested part of calculated digest. |
| GetDigestService | Get the [DigestService](#_bookmark161) instance. |
| Start | Initialize the context for a new data stream processing or generation (depending on the primitive). |
| Update | Update the digest calculation context by a new part of the message. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | MessageAuthnCodeCtx | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Keyed message authentication code context interface definition (MAC/HMAC). | |
| ***Operations:*** | Check | Check the calculated digest against an expected "signature" object. |
| Finish | Finish the digest calculation and optionally produce the "signature" object. |
| GetDigest | Get requested part of calculated digest to existing memory buffer. |
| GetDigestService | Get the [DigestService](#_bookmark161) instance. |
| Reset | Clear the crypto context. |
| SetKey | Set (deploy) a key to the message authn code algorithm context. |
| Start | Initialize the context for a new data stream processing or generation (depending from the primitive). |
| Update | Update the digest calculation context by a new part of the message. |



Adaptive Application

«use»



*CryptoContext CryptoContext CryptoContext*

«aapAPI,aapNativeInterf... «aapAPI,aapNativeInterf... «aapAPI,aapNativeInterface» KeyEncapsulatorPublicCtx KeyDecapsulatorPrivateCtx SymmetricKeyWrapperCtx

*CryptoContext*

«aapAPI,aapNativeInt... RandomGeneratorCtx

«aapAPI,aapNativeInterf...

ExtensionService

+ AddKeyingData()

+ Encapsulate()

+ GetEncapsulatedSize()

+ GetExtensionService()

+ GetKekEntropy()

+ Reset()

+ SetKey()

+ DecapsulateKey()

+ DecapsulateSeed()

+ GetEncapsulatedSize()

+ GetExtensionService()

+ GetKekEntropy()

+ Reset()

+ SetKey()

+ CalculateWrappedKeySize()

+ GetExtensionService()

+ GetMaxTargetKeyLength()

+ GetTargetKeyGranularity()

+ Reset()

+ SetKey()

+ UnwrapConcreteKey()

+ UnwrapKey()

+ UnwrapSeed()

+ WrapKeyMaterial()

+ AddEntropy()

+ Generate()

+ GetExtensionService()

+ Seed()

+ SetKey()

+ GetActualKeyBitLength()

+ GetActualKeyCOUID()

+ GetAllowedUsage()

+ GetMaxKeyBitLength()

+ GetMinKeyBitLength()

+ IsKeyAvailable()

+ IsKeyBitLengthSupported()

«aapFunctionalCluster» Cryptography

daemon-based

«use»

«use»

«use»

«use»

**Figure 9.48: ExtensionService and CryptoContext Interfaces (1 of 2)**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | ExtensionService | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Basic meta-information service for all contexts. | |
| ***Operations:*** | GetActualKeyBitLength | Get actual bit-length of a key loaded to the context. |
| GetActualKeyCOUID | Get the CryptoObjectUid of the key deployed to the context this extension service is attached to. |
| GetAllowedUsage | Get allowed usages of this context (according to the key object attributes loaded to this context). |
| GetMaxKeyBitLength | Get maximum supported key length in bits. |
| GetMinKeyBitLength | Get minimal supported key length in bits. |
| IsKeyAvailable | Check if a key has been set to this context. |
| IsKeyBitLengthSupported | Verify if the specific key length is supported by the context. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | KeyEncapsulatorPublicCtx | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Asymmetric key encapsulation mechanism public key context interface. | |
| ***Operations:*** | AddKeyingData | Add the content to be encapsulated (payload) according to RFC 5990 ("keying data"). |
| Encapsulate | Encapsulate the last set keying-data. |
| GetEncapsulatedSize | Get fixed size of the encapsulated data block. |
| GetExtensionService | Get the [ExtensionService](#_bookmark162) instance. |
| GetKekEntropy | Get entropy (bit-length) of the key encryption key (KEK) material. |
| Reset | Clear the crypto context. |

*Δ*

|  |  |  |
| --- | --- | --- |
|  | SetKey | Set (deploy) a key to the key encapsulator public algorithm context. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | KeyDecapsulatorPrivateCtx | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Asymmetric key encapsulation mechanism private key context interface. | |
| ***Operations:*** | DecapsulateKey | Decapsulate the keying data to be used for subsequent processing (e.g. secure communication). |
| DecapsulateSeed | Decapsulate key material. |
| GetEncapsulatedSize | Get fixed size of the encapsulated data block. |
| GetExtensionService | Get the [ExtensionService](#_bookmark162) instance. |
| GetKekEntropy | Get entropy (bit-length) of the key encryption key (KEK) material. |
| Reset | Clear the crypto context. |
| SetKey | Set (deploy) a key to the key decapsulator private algorithm context. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | SymmetricKeyWrapperCtx | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Context of a symmetric key wrap algorithm (for AES it should be compatible with RFC3394 or RFC5649). | |
| ***Operations:*** | CalculateWrappedKeySize | Calculate size of the wrapped key in bytes from original key length in bits. |
| GetExtensionService | Get the [ExtensionService](#_bookmark162) instance. |
| GetMaxTargetKeyLength | Get maximum length of the target key supported by the implementation. |
| GetTargetKeyGranularity | Get expected granularity of the target key (block size). |
| Reset | Clear the crypto context. |
| SetKey | Set (deploy) a key to the symmetric key wrapper algorithm context. |
| UnwrapConcreteKey | Execute the "key unwrap" operation for the provided BLOB and produce a key object of the expected type. |
| UnwrapKey | Execute the "key unwrap" operation for the provided BLOB and produce a key object. |
| UnwrapSeed | Execute the "key unwrap" operation for the provided BLOB and produce a [SecretSeed](#_bookmark165) object. |
| WrapKeyMaterial | Execute the "key wrap" operation for the provided key material. |

|  |  |
| --- | --- |
| ***Name:*** | RandomGeneratorCtx |
| ***Technology:*** | Native interface |
| ***Usage:*** | Public API |
| ***Description:*** | Interface of a random number generator context. |

*Δ*

|  |  |  |
| --- | --- | --- |
| ***Operations:*** | AddEntropy | Update the internal state of the RNG by mixing it with the provided additional entropy. |
| Generate | Return an allocated buffer with a generated random sequence of the requested size. |
| GetExtensionService | Get the [ExtensionService](#_bookmark162) instance. |
| Seed | Set the internal state of the RNG using the provided seed. |
| SetKey | Set the internal state of the RNG using the provided seed. |



Adaptive Application

«use»



*CryptoContext*

«aapAPI,aapNativeInterface» KeyDerivationFunctionCtx

*CryptoContext*

«aapAPI,aapNativeInterf... KeyAgreementPrivateCtx

*CryptoContext*

«aapAPI,aapNativeInt... MsgRecoveryPublicCtx

*CryptoContext*

«aapAPI,aapNativeInt... SigEncodePrivateCtx

«aapAPI,aapNativeInterf...

ExtensionService

+ AddSalt()

+ AddSecretSalt()

+ ConfigIterations()

+ DeriveKey()

+ DeriveSeed()

+ GetExtensionService()

+ GetKeyIdSize()

+ GetTargetAlgId()

+ GetTargetAllowedUsage()

+ GetTargetKeyBitLength()

+ Init()

+ Reset()

+ SetSourceKeyMaterial()

+ AgreeKey()

+ AgreeSeed()

+ GetExtensionService()

+ Reset()

+ SetKey()

+ DecodeAndVerify()

+ GetExtensionService()

+ GetMaxInputSize()

+ GetMaxOutputSize()

+ Reset()

+ SetKey()

+ GetExtensionService()

+ GetMaxInputSize()

+ GetMaxOutputSize()

+ Reset()

+ SetKey()

+ SignAndEncode()

+ GetActualKeyBitLength()

+ GetActualKeyCOUID()

+ GetAllowedUsage()

+ GetMaxKeyBitLength()

+ GetMinKeyBitLength()

+ IsKeyAvailable()

+ IsKeyBitLengthSupported()

«aapFunctionalCluster» Cryptography

daemon-based

«use»

«use»

«use»

«use»

**Figure 9.49: ExtensionService and CryptoContext Interfaces (2 of 2)**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | KeyDerivationFunctionCtx | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Key derivation function interface. | |
| ***Operations:*** | AddSalt | Add a salt value stored in a non-secret memory region. |
| AddSecretSalt | Add a secret salt value stored in a [SecretSeed](#_bookmark165) object. |
| ConfigIterations | Configure the number of iterations that will be applied by default. |
| DeriveKey | Derive a symmetric key from the provided key material and provided context configuration. |
| DeriveSeed | Derive key material (secret seed) from the provided "master" key material and the provided context configuration. |
| GetExtensionService | Get the [ExtensionService](#_bookmark162) instance. |
| GetKeyIdSize | Get the fixed size of the target key ID required by diversification algorithm. |
| GetTargetAlgId | Get the symmetric algorithm ID of target key. |
| GetTargetAllowedUsage | Get allowed key usage of target key. |

*Δ*

|  |  |  |
| --- | --- | --- |
|  | GetTargetKeyBitLength | Get the bit-length of target (diversified) keys. |
| Init | Initialize this context by setting at least the target key ID. |
| Reset | Clear the crypto context. |
| SetSourceKeyMaterial | Set (deploy) key-material to the key derivation algorithm context. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | KeyAgreementPrivateCtx | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Key agreement private key context interface (Diffie Hellman or conceptually similar). | |
| ***Operations:*** | AgreeKey | Produce a common symmetric key via execution of the key-agreement algorithm between this private key. |
| AgreeSeed | Produce a common secret seed via execution of the key-agreement algorithm between this private key. |
| GetExtensionService | Get the [ExtensionService](#_bookmark162) instance. |
| Reset | Clear the crypto context. |
| SetKey | Set (deploy) a key to the key agreement private algorithm context. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | MsgRecoveryPublicCtx | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | A public key context for asymmetric recovery of a short message and its signature verification (RSA-like). | |
| ***Operations:*** | DecodeAndVerify | Process (encrypt / decrypt) an input block according to the cryptor configuration. |
| GetExtensionService | Get the [ExtensionService](#_bookmark162) instance. |
| GetMaxInputSize | Get maximum expected size of the input data block. |
| GetMaxOutputSize | Get maximum possible size of the output data block. |
| Reset | Clear the crypto context. |
| SetKey | Set (deploy) a key to the msg recovery public algorithm context. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | SigEncodePrivateCtx | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | A private key context for asymmetric signature calculation and short message encoding (RSA-like). | |
| ***Operations:*** | GetExtensionService | Get the [ExtensionService](#_bookmark162) instance. |
| GetMaxInputSize | Get maximum expected size of the input data block. |
| GetMaxOutputSize | Get maximum possible size of the output data block. |
| Reset | Clear the crypto context. |
| SetKey | Set (deploy) a key to the sig encode private algorithm context. |
| SignAndEncode | Process (encrypt / decrypt) an input block according to the cryptor configuration. |



Adaptive Application

«use» «use» «use»



daemon-based

«aapFunctionalCluster» Cryptography

+ GetRequiredHashAlgId()

+ GetRequiredHashSize()

+ GetSignatureSize()

+ GetSignatureService()

+ Verify()

+ VerifyPrehashed()

+ GetSignatureService()

+ Reset()

+ SetKey()

+ Sign()

+ SignPreHashed()

*ExtensionService*

«aapAPI,aapNativeInterface» SignatureService

*CryptoContext*

«aapAPI,aapNativeInterface» VerifierPublicCtx

*CryptoContext*

«aapAPI,aapNativeInterface» SignerPrivateCtx

**Figure 9.50: SignatureService and CryptoContext Interfaces**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | SignatureService | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Extension meta-information service for signature contexts. | |
| ***Operations:*** | GetRequiredHashAlgId | Get an ID of hash algorithm required by current signature algorithm. |
| GetRequiredHashSize | Get the hash size required by current signature algorithm. |
| GetSignatureSize | Get size of the signature value produced and required by the current algorithm. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | SignerPrivateCtx | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Signature private key context interface. | |
| ***Operations:*** | GetSignatureService | Get the [SignatureService](#_bookmark163) instance. |
| Reset | Clear the crypto context. |
| SetKey | Set (deploy) a key to the signer private algorithm context. |
| Sign | Sign a directly provided hash or message value. |
| SignPreHashed | Sign a provided digest value stored in the hash-function context. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | VerifierPublicCtx | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Signature verification public key context interface. | |
| ***Operations:*** | GetSignatureService | Get the [SignatureService](#_bookmark163) instance. |
| Verify | Verify signature BLOB by a directly provided hash or message value. |

*Δ*

|  |  |  |
| --- | --- | --- |
|  | VerifyPrehashed | Verify a signature by a digest value stored in the hash-function context. |

* + - * 1. **Cryptographic object interfaces**

*RestrictedUseObject*

«aapAPI,aapNativeInte...

SymmetricKey



Adaptive Application

«use»

«use»

«use»



daemon-based

«aapFunctionalCluster» Cryptography

+ GetPublicKey()

+ GetPrimitiveId()

+ GetPrimitiveName()

*RestrictedUseObject*

«aapAPI,aapNativeInterf...

PrivateKey

«aapAPI,aapNativeInter...

CryptoPrimitiveId

**Figure 9.51: Cryptographic Object Interfaces**



«aapAPI,aapNativeInterf...

Signature

«aapAPI,aapNativeInterf...

PublicKey

«aapAPI,aapNativeInterf...

PrivateKey

«aapAPI,aapNativeInterf...

SecretSeed

«aapAPI,aapNativeInterf...

SymmetricKey

«aapAPI,aapNativeInterf...

RestrictedUseObject

«aapAPI,aapNativeInterface» Serializable

«aapAPI,aapNativeInterf...

CryptoObject

**Figure 9.52: Taxonomy of Cryptographic Object Interfaces**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | CryptoObject | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | A common interface for all cryptographic objects recognizable by the [CryptoProvider](#_bookmark157). | |
| ***Operations:*** | GetCryptoPrimitiveId | Return the [CryptoPrimitiveId](#_bookmark164) of this object. |
| GetObjectId | Return the object’s COIdentifier, which includes the object’s type and UID. |
| GetPayloadSize | Return actual size of the object’s payload. |
| HasDependence | Return the COIdentifier of the object that this object depends on. |
| IsExportable | Get the exportability attribute of the crypto object. |

*Δ*

|  |  |  |
| --- | --- | --- |
|  | IsSession | Return the session (or temporary) attribute of the object. |
| Save | Save itself to provided [IOInterface](#_bookmark155) |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | RestrictedUseObject | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | A common interface for all objects supporting the usage restriction. | |
| ***Operations:*** | GetAllowedUsage | Get allowed usages of this object. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | CryptoPrimitiveId | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Common interface for identification of all CryptoPrimitives and their keys and parameters. | |
| ***Operations:*** | GetPrimitiveId | Get vendor specific ID of the primitive. |
| GetPrimitiveName | Get a unified name of the primitive. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | SecretSeed | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Secret seed object contains a raw bit sequence of specific length (without any filtering of allowed/ disallowed values). | |
| ***Operations:*** | Clone | Clone this object to new session object. |
| Jump | Set value of this seed object as a "jump" from it’s current state to specified number of steps, according to "counting" expression defined by a cryptographic algorithm associated with this object. |
| JumpFrom | Set value of this seed object as a "jump" from an initial state to specified number of steps, according to "counting" expression defined by a cryptographic algorithm associated with this object. |
| Next | Set next value of the secret seed according to "counting" expression defined by a cryptographic algorithm associated with this object. |
| operator ˆ= | XOR value of this seed object with another one and save result to this object. If seed sizes in this object and in the source argument are different then only correspondent number of leading bytes in this seed object shall be updated. |

|  |  |
| --- | --- |
| ***Name:*** | SymmetricKey |
| ***Technology:*** | Native interface |
| ***Usage:*** | Public API |
| ***Description:*** | Symmetric Key interface. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | PublicKey | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | General asymmetric public key interface. | |
| ***Operations:*** | CheckKey | Check the key for its correctness. |
| HashPublicKey | Calculate hash of the public key value. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | PrivateKey | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Generalized asymmetric private key interface. | |
| ***Operations:*** | GetPublicKey | Get the public key correspondent to this private key. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | Signature | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface is applicable for keeping the Digital Signature, Hash Digest, (Hash-based) Message Authentication Code (MAC/HMAC). | |
| ***Operations:*** | GetHashAlgId | Get an ID of hash algorithm used for this signature object production. |
| GetRequiredHashSize | Get the hash size required by current signature algorithm. |

* + - * 1. **Cryptographic key handling interfaces**

Adaptive Application



«use» «use»

«aapRequiredPort»

«aapAPI,aapNativeInterface» KeyStorageProvider

+ BeginTransaction()

+ CommitTransaction()

+ GetRegisteredObserver()

+ LoadKeySlot()

+ RegisterObserver()

+ RollbackTransaction()

+ UnsubscribeObserver()

«aapAPI,aapPortInterface» KeySlot

+ Clear()

+ GetContentProps()

+ GetPrototypedProps()

+ IsEmpty()

+ MyProvider()

+ Open()

+ SaveCopy()

«aapAPI,aapNativeInterface» UpdatesObserver

«aapCallbackMethod»

+ OnUpdate()

«use»

«aapFunctionalCluster»

Cryptography

daemon-based

**Figure 9.53: Cryptographic Key Handling Interfaces**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | KeyStorageProvider | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Key Storage Provider interface. | |
| ***Operations:*** | BeginTransaction | Begin new transaction for key slots update. |
| CommitTransaction | Commit changes of the transaction to storage. |
| GetRegisteredObserver | Get the currently registered [UpdatesObserver](#_bookmark171) for key slots. |
| LoadKeySlot | Load a key slot. |
| RegisterObserver | Register an [UpdatesObserver](#_bookmark171) for key slots. |
| RollbackTransaction | Rollback all changes executed during the transaction in storage. |
| UnsubscribeObserver | Unregister an [UpdatesObserver](#_bookmark171) from key slots. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | KeySlot | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | CryptoKeySlot | |
| ***Usage:*** | Public API | |
| ***Description:*** | Key slot interface enables access to a physical key-slot. | |
| ***Operations:*** | Clear | Clear the content of this key slot. |
| GetContentProps | Get an actual properties of a content in the key slot. |
| GetPrototypedProps | Get the prototyped properties of the key slot. |
| IsEmpty | Check the slot for emptiness. |
| MyProvider | Retrieve the instance of the [CryptoProvider](#_bookmark157) that owns this [KeySlot](#_bookmark170). |
| Open | Open this key slot and return an [IOInterface](#_bookmark155) to its content. |
| SaveCopy | Save the content of a provided source [IOInterface](#_bookmark155) to this key slot. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | UpdatesObserver | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Interface for observing updates on key slots. | |
| ***Operations:*** | OnUpdate | This method is called if the content of the specified slots was changed. |

* + - * 1. **X.509 certificate handling interfaces**



Adaptive Application

«use»

«aapAPI,aapNativeInterface» X509Provider

«aapAPI,aapNativeInterface» X509CustomExtensionsParser

+ BuildDn()

+ CheckCertStatus()

+ CheckCertStatusOnline()

+ CleanupVolatileStorage()

+ CountCertsInChain()

+ CreateCertSignRequest()

+ CreateEmptyDn()

+ CreateEmptyExtensions()

+ CreateOcspRequest()

+ DecodeDn()

+ FindCertByDn()

+ FindCertByKeyIds()

+ FindCertBySn()

+ Import()

+ ImportCrl()

+ LoadCertificate()

+ ParseCert()

+ ParseCertChain()

+ ParseCertSignRequest()

+ ParseCustomCertExtensions()

+ ParseOcspResponse()

+ Remove()

+ SendRequest()

+ SetAsRootOfTrust()

+ SetPendingStatus()

+ UpdateCrlOnline()

+ VerifyCert()

+ VerifyCertChain()

+ OnBitString()

+ OnBool()

+ OnGeneralizedTime()

+ OnIa5String()

+ OnInteger()

+ OnNull()

+ OnOctetString()

+ OnOid()

+ OnParsingEnd()

+ OnPrintableString()

+ OnSequenceEnd()

+ OnSequenceStart()

+ OnSetEnd()

+ OnSetStart()

+ OnUtcTime()

+ OnUtf8String()

«use»

«aapFunctionalCluster» Cryptography

daemon-based

**Figure 9.54: X.509 Certificate Handling Interfaces**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | X509Provider | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | X.509 Provider interface supporting two internal storage types: volatile (or session) and persistent. | |
| ***Operations:*** | BuildDn | Create completed X.500 Distinguished Name structure from the provided string representation. |
| CheckCertStatus | Check certificate status by directly provided OCSP response. |
| CheckCertStatusOnline | Check certificate status via On-line Certificate Status Protocol (OCSP). |
| CleanupVolatileStorage | Cleanup the volatile certificates storage. |
| CountCertsInChain | Count number of certificates in a serialized certificate chain represented by a single BLOB. |
| CreateCertSignRequest | Create certification request for a private key loaded to the context. |
| CreateEmptyDn | Create an empty X.500 Distinguished Name (DN) structure. |
| CreateEmptyExtensions | Create an empty X.509 Extensions structure. |
| CreateOcspRequest | Create OCSP request for specified certificate(s). |

*Δ*

|  |  |  |
| --- | --- | --- |
|  | DecodeDn | Decode X.500 Distinguished Name structure from the provided serialized format. |
| FindCertByDn | Find a certificate by the subject and issuer Distinguished Names (DN). |
| FindCertByKeyIds | Find a certificate by its SKID & AKID. |
| FindCertBySn | Find a certificate by its serial number and issue DN. |
| Import | Import the certificate to volatile or persistent storage. |
| ImportCrl | Import Certificate Revocation List (CRL) or Delta CRL from a memory BLOB. |
| LoadCertificate | Load a certificate from the persistent certificate storage. |
| ParseCert | Parse a serialized representation of the certificate and create its instance. |
| ParseCertChain | Parse a serialized representation of the certificate chain and create their instances. |
| ParseCertSignRequest | Parse a certificate signing request (CSR) provided by the user. |
| ParseCustomCertExtensions | Parse the custom X.509 extensions. |
| ParseOcspResponse | Parse serialized OCSP response and create correspondent interface instance. |
| Remove | Remove specified certificate from the storage (volatile or persistent) and destroy it. |
| SendRequest | Send prepared certificate request to CA and save it to volatile or persistent storage. |
| SetAsRootOfTrust | Set specified CA certificate as a "root of trust". |
| SetPendingStatus | Set the "pending" status associated to the CSR that means that the CSR already sent to CA. |
| UpdateCrlOnline | Get Certificate Revocation List (CRL) or Delta CRL via on-line connection. |
| VerifyCert | Verify status of the provided certificate by locally stored CA certificates and CRLs only. |
| VerifyCertChain | Verify status of the provided certification chain by locally stored CA certificates and CRLs only. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | X509CustomExtensionsParser | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | X.509 custom extensions parser. This callback interface is to be implemented by an application. | |
| ***Operations:*** | OnBitString | Called when a bit string is encountered. |
| OnBool | Called when a boolean is encountered. |
| OnGeneralizedTime | Called when a generalized time is encountered. |
| OnIa5String | Called when an IA5 string is encountered. |
| OnInteger | Called when an integer is encountered. |
| OnNull | Called when a NULL is encountered. |
| OnOctetString | Called when an octet string is encountered. |
| OnOid | Called when an oid is encountered. |
| OnParsingEnd | Called when the parsing is completed. |
| OnPrintableString | Called when a printable string is encountered. |
| OnSequenceEnd | Called when a sequence ends. |

*Δ*

|  |  |  |
| --- | --- | --- |
|  | OnSequenceStart | Called when a sequence starts. |
| OnSetEnd | Called when a set ends. |
| OnSetStart | Called when a set starts. |
| OnUtcTime | Called when a UTC time is encountered. |
| OnUtf8String | Called when an UTF8 string is encountered. |



Adaptive Application

«use»

«use»

«use»

«use»

*BasicCertInfo*



«aapAPI,aapPortInterface» Certificate

*BasicCertInfo*

«aapAPI,aapNativeInterface» CertSignRequest

*X509Object*

«aapAPI,aapNativeInterface» X509DN

*Serializable*

«aapAPI,aapNativeInterface» X509PublicKeyInfo

+ AuthorityKeyId()

+ EndTime()

+ GetFingerprint()

+ GetStatus()

+ IsRoot()

+ IssuerDn()

+ SerialNumber()

+ StartTime()

+ SubjectKeyId()

+ VerifyMe()

+ X509Version()

+ ExportASN1CertSignRequest()

+ GetSignature()

+ Verify()

+ Version()

+ GetAttribute()

+ GetDnString()

+ SetAttribute()

+ SetDn()

+ GetAlgorithmId()

+ GetPublicKey()

+ GetRequiredHashAlgId()

+ GetRequiredHashSize()

+ GetSignatureSize()

+ IsSameKey()

daemon-based

«aapFunctionalCluster» Cryptography

**Figure 9.55: X.509 Certificate Object Interfaces**



«aapAPI,aapNativeInt...

X509PublicKeyInfo

«aapAPI,aapNativeInt...

X509Object

«aapAPI,aapNativeInt...

Serializable



«aapAPI,aapNativeInt...

X509DN

«aapAPI,aapNativeInt...

OcspResponse

«aapAPI,aapNativeInt...

OcspRequest

**Figure 9.56: Taxonomy of X.509 Certificate Object Interfaces**



«aapAPI,aapNativeInt...

CertSignRequest

«aapAPI,aapPortInterf...

Certificate

«aapAPI,aapNativeInt...

BasicCertInfo

«aapAPI,aapNativeInt...

X509Extensions

|  |  |  |
| --- | --- | --- |
| ***Name:*** | OcspRequest | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | On-line Certificate Status Protocol Request. | |
| ***Operations:*** | Version | Get version of the OCSP request format. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | OcspResponse | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | On-line Certificate Status Protocol Response. | |
| ***Operations:*** | Version | Get version of the OCSP response format. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | Certificate | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | X.509 Certificate interface. | |
| ***Operations:*** | AuthorityKeyId | Get the DER encoded AuthorityKeyIdentifier of this certificate. |
| EndTime | Get the NotAfter of the certificate. |
| GetFingerprint | Calculate a fingerprint from the whole certificate. |
| GetStatus | Return last verification status of the certificate. |
| IsRoot | Check whether this certificate belongs to a root CA. |
| IssuerDn | Get the issuer certificate DN. |
| SerialNumber | Get the serial number of this certificate. |
| StartTime | Get the NotBefore of the certificate. |
| SubjectKeyId | Get the DER encoded SubjectKeyIdentifier of this certificate. |
| VerifyMe | Verify signature of the certificate. |
| X509Version | Get the X.509 version of this certificate object. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | CertSignRequest | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Certificate Signing Request (CSR) object interface. | |
| ***Operations:*** | ExportASN1CertSignRequest | Export this certificate signing request in DER encoded ASN1 format. |
| GetSignature | Return signature object of the request. |
| Verify | Verifies self-signed signature of the certificate request. |
| Version | Return format version of the certificate request. |

|  |  |
| --- | --- |
| ***Name:*** | X509Extensions |
| ***Technology:*** | Native interface |
| ***Usage:*** | Public API |

*q*

*Δ*

|  |  |  |
| --- | --- | --- |
| ***Description:*** | Interface of X.509 Extensions. | |
| ***Operations:*** | Count | Count number of elements in the sequence. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | X509DN | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Interface of X.509 Distinguished Name (DN). | |
| ***Operations:*** | GetAttribute | Get a DN attribute. |
| GetDnString | Get the whole Distinguished Name (DN) as a single string. |
| SetAttribute | Set a DN attribute. |
| SetDn | Set whole Distinguished Name (DN) from a single string. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | X509PublicKeyInfo | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | X.509 Public Key Information interface. | |
| ***Operations:*** | GetAlgorithmId | Get the CryptoPrimitiveId instance of this class. |
| GetPublicKey | Get public key object of the subject. |
| GetRequiredHashAlgId | Get an ID of hash algorithm required by current signature algorithm. |
| GetRequiredHashSize | Get the hash size required by current signature algorithm. |
| GetSignatureSize | Get size of the signature value produced and required by the current algorithm. |
| IsSameKey | Verify the sameness of the provided and kept public keys. |

* + - 1. **Provided interfaces**



«aapFunctionalCluster» Persistency



UCM Subordinate



«aapFunctionalClust...

Communication Management

«use» «use» «use»

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| «aapAPI,aapNativeInterface» CryptoStack | | | |
|  | |  | |
| daemon-based | «aapFunctionalCluster» Cryptography | |  |

**Figure 9.57: Users of the Cryptography interfaces**

|  |  |
| --- | --- |
| ***Interface*** | ***Requiring functional clusters*** |
| [Cryptography](#_bookmark150)::[CryptoStack](#_bookmark151) | [Communication Management](#_bookmark105) |
| [Persistency](#_bookmark124) |
| [Update and Configuration Management](#_bookmark200) |

**Table 9.16: Interfaces provided by Cryptography to other Functional Clusters**

* + - 1. **Required interfaces**



«use»



«use»

«aapRequiredPort»

«aapFunctionalCl... Platform Health Management

daemon-based

+ ReportCheckpoint()

«aapAPI,aapPortInterf...

SupervisedEntity

«aapFunctionalCluster»

Cryptography

daemon-based

«use»



«aapFunctionalCluster» Identity and Access Management

+ CheckAccess()

«aapInternal,aapNativeInterf...

PolicyDecisionPoint



«aapFunctionalClust...

Log and Trace

+ IsEnabled()

+ Log(MsgId, Params)

+ LogDebug(): LogStream

+ LogError(): LogStream

+ LogFatal(): LogStream

+ LogInfo(): LogStream

+ LogVerbose(): LogStream

+ LogWarn(): LogStream

+ WithLevel(): LogStream

«aapAPI,aapNativeInterf...

Logger

**Figure 9.58: Interfaces required by Cryptography**

|  |  |
| --- | --- |
| ***Interface*** | ***Purpose*** |
| [Adaptive Intrusion Detection System Manager](#_bookmark178):: [EventReporter](#_bookmark179) | This interface should be used to e.g., report attempts to change root certificates. |
| [Cryptography](#_bookmark150)::[UpdatesObserver](#_bookmark171) | [Cryptography](#_bookmark150) uses this interface to notify when a key has been updated. |
| [Cryptography](#_bookmark150)::[X509CustomExtensionsParser](#_bookmark173) | [Cryptography](#_bookmark150) uses this interface for propagating parser events. |
| [Identity and Access Management](#_bookmark175)::[Policy](#_bookmark176) [DecisionPoint](#_bookmark176) | [Cryptography](#_bookmark150) shall use this interface to check access to certificates. |
| [Log and Trace](#_bookmark90)::[Logger](#_bookmark91) | [Cryptography](#_bookmark150) shall use this interface to log standardized messages. |
| [Platform Health Management](#_bookmark186)::[SupervisedEntity](#_bookmark189) | [Cryptography](#_bookmark150) should use this interface for supervision of its daemon process(es). |
| [Registry](#_bookmark210)::[ManifestAccessor](#_bookmark211) | [Cryptography](#_bookmark150) shall use this interface to read its configuration information from the Manifests. |

**Table 9.17: Interfaces required by Cryptography**

### Identity and Access Management

|  |  |
| --- | --- |
| ***Name:*** | Identity and Access Management |
| ***Short Name:*** | iam |
| ***Category:*** | Security |
| ***Daemon-based:*** | No |
| ***Responsibilities:*** | [Identity and Access Management](#_bookmark175) checks access to resources of the AUTOSAR Adaptive Platform, for example, on Service Interfaces and Functional Clusters. [Identity and](#_bookmark175) [Access Management](#_bookmark175) hereby introduces access control for [Adaptive Application](#_bookmark24)s and protection against privilege escalation in case of attacks. In addition, [Identity and Access](#_bookmark175) [Management](#_bookmark175) enables integrators to verify access on resources requested by [Adaptive](#_bookmark24) [Application](#_bookmark24)s in advance during deployment. |

* + - 1. **Defined interfaces**



«aapFunctionalCluster» Identity and Access Management

+ CheckAccess()

«aapInternal,aapNativeInterface» PolicyDecisionPoint

**Figure 9.59: Interfaces of Identity and Access Management**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | PolicyDecisionPoint | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Internal | |
| ***Description:*** | This interface serves Policy Enforcement Points with authorization decisions. | |
| ***Operations:*** | CheckAccess | Evaluates an access request against the authorization policies (Grant) before issuing an access decision. |

* + - 1. **Provided interfaces**



«aapFunctionalClust...

Cryptography daemon-based



«aapFunctionalClust...

Communication Management



«aapFunctionalClust... Diagnostic Management

daemon-based

«use»



«aapFunctionalCluster» Identity and Access Management

+ CheckAccess()

«aapInternal,aapNativeInterface» PolicyDecisionPoint

«use»

«use»

**Figure 9.60: Users of the Identity and Access Management interfaces**

|  |  |
| --- | --- |
| ***Interface*** | ***Requiring functional clusters*** |
| [Identity and Access Management](#_bookmark175)::[Policy](#_bookmark176) [DecisionPoint](#_bookmark176) | [Communication Management](#_bookmark105) |
| [Cryptography](#_bookmark150) |
| [Diagnostic Management](#_bookmark214) |

**Table 9.18: Interfaces provided by Identity and Access Management to other Functional Clusters**

* + - 1. **Required interfaces**

«use» «use»



«aapFunctionalCluster» Log and Trace

+ IsEnabled()

+ Log(MsgId, Params)

+ LogDebug(): LogStream

+ LogError(): LogStream

+ LogFatal(): LogStream

+ LogInfo(): LogStream

+ LogVerbose(): LogStream

+ LogWarn(): LogStream

+ WithLevel(): LogStream

«aapAPI,aapNativeInterfa...

Logger



«aapFunctionalCluster» Registry

«aapInternal,aapNativeInt...

ManifestAccessor



«aapFunctionalCluster» Identity and Access Management

**Figure 9.61: Interfaces required by Identity and Access Management**

|  |  |
| --- | --- |
| ***Interface*** | ***Purpose*** |
| [Adaptive Intrusion Detection System Manager](#_bookmark178):: [EventReporter](#_bookmark179) | This interface should be used to e.g., report denied access. |
| [Log and Trace](#_bookmark90)::[Logger](#_bookmark91) | [Identity and Access Management](#_bookmark175) shall use this interface to log standardized messages. |
| [Registry](#_bookmark210)::[ManifestAccessor](#_bookmark211) | [Identity and Access Management](#_bookmark175) should use this interface to access Grant information modeled in the Manifests. |

**Table 9.19: Interfaces required by Identity and Access Management**

### Adaptive Intrusion Detection System Manager

|  |  |
| --- | --- |
| ***Name:*** | Adaptive Intrusion Detection System Manager |
| ***Short Name:*** | idsm |
| ***Category:*** | Security |

*q*

*Δ*

|  |  |
| --- | --- |
| ***Daemon-based:*** | Yes |
| ***Responsibilities:*** | [Adaptive Intrusion Detection System Manager](#_bookmark178) provides functionality to report security events. |

* + - 1. **Defined interfaces**



Adaptive Application

«use» «use»



«aapFunctionalCluster»

Adaptive Intrusion Detection System Manager daemon-based

+ ReportEvent()

+ RegisterTimestampProvider()

«aapNativeInterface,aapAPI»

«aapPortInterface,aapAPI» EventReporter

**Figure 9.62: Interfaces of Adaptive Intrusion Detection System Manager**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | EventReporter | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | SecurityEventDefinition | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface is used to report security events to the [Adaptive Intrusion Detection](#_bookmark178) [System Manager](#_bookmark178). | |
| ***Operations:*** | ReportEvent | Create a new security event at the [Adaptive](#_bookmark178) [Intrusion Detection System Manager](#_bookmark178). |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | TimestampProvider | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Provides functionality to register a timestamp provider with the [Adaptive Intrusion](#_bookmark178) [Detection System Manager](#_bookmark178). | |
| ***Operations:*** | RegisterTimestampProvider | Register a callback for providing timestamps to the [Adaptive Intrusion Detection System](#_bookmark178) [Manager](#_bookmark178). |

* + - 1. **Provided interfaces**



«aapFunctionalClust...

Communication Management

«use»



«aapFunctionalClust...

Cryptography daemon-based



«aapFunctionalClust...

Firewall daemon-based

«aapFunctionalClust... Identity and Access

Management



«aapFunctionalClust...

Execution Management daemon-based

«use» «use»

«use» «use»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| «aapPortInterface,aapAPI» EventReporter | | | | | |
| + ReportEvent() | | | | | |
|  | | |  | | |
| «aapFunctionalCluster»  Adaptive Intrusion Detection System Manager  daemon-based | | | | | |

**Figure 9.63: Users of the Adaptive Intrusion Detection System Manager interfaces**

|  |  |
| --- | --- |
| ***Interface*** | ***Requiring functional clusters*** |
| [Adaptive Intrusion Detection System Manager](#_bookmark178):: [EventReporter](#_bookmark179) | [Communication Management](#_bookmark105) |
| [Cryptography](#_bookmark150) |
| [Execution Management](#_bookmark61) |
| [Firewall](#_bookmark182) |
| [Identity and Access Management](#_bookmark175) |

**Table 9.20: Interfaces provided by Adaptive Intrusion Detection System Manager to other Functional Clusters**

* + - 1. **Required interfaces**

«aapFunctionalCluster»

Adaptive Intrusion Detection System Manager

daemon-based

«use»



«aapRequiredPort»

«use» «use»



«aapAPI,aapPortInterface» SynchronizedTimeBaseConsumer

+ GetCurrentTime()

+ GetRateDeviation()

+ GetTimeWithStatus(): SynchronizedTimeBaseStatus

+ RegisterStatusChangeNotifier()

+ RegisterSynchronizationStateChangeNotifier()

+ RegisterTimeLeapNotifier()

+ RegisterTimePrecisionMeasurementNotifier()

+ RegisterTimeValidationNotification()

+ UnregisterStatusChangeNotifier()

+ UnregisterSynchronizationStateChangeNotifier()

+ UnregisterTimeLeapNotifier()

+ UnregisterTimePrecisionMeasurementNotifier()

+ UnregisterTimeValidationNotification()

«aapFunctionalCluster»

Time Synchronization

daemon-based



«aapAPI,aapNativeInterf...

Logger

+ IsEnabled()

+ Log(MsgId, Params)

+ LogDebug(): LogStream

+ LogError(): LogStream

+ LogFatal(): LogStream

+ LogInfo(): LogStream

+ LogVerbose(): LogStream

+ LogWarn(): LogStream

+ WithLevel(): LogStream

«aapFunctionalClust...

Log and Trace

**Figure 9.64: Interfaces required by Adaptive Intrusion Detection System Manager**



«aapInternal» TCP/IP Stack

Operating System

|  |  |
| --- | --- |
| ***Interface*** | ***Purpose*** |
| TCP/IP Stack | [Adaptive Intrusion Detection System Manager](#_bookmark178) shall use this interface to propagate qualified security events via the IDS protocol. |
| [Log and Trace](#_bookmark90)::[Logger](#_bookmark91) | [Adaptive Intrusion Detection System Manager](#_bookmark178) shall use this interface to log standardized messages. |
| [Time Synchronization](#_bookmark115)::[SynchronizedTimeBase](#_bookmark120) [Consumer](#_bookmark120) | [Adaptive Intrusion Detection System Manager](#_bookmark178) shall use this interface to determine timestamps of security events. |

**Table 9.21: Interfaces required by Adaptive Intrusion Detection System Manager**

### Firewall

|  |  |
| --- | --- |
| ***Name:*** | Firewall |
| ***Short Name:*** | fw |
| ***Category:*** | Security |
| ***Daemon-based:*** | Yes |
| ***Responsibilities:*** | The [Firewall](#_bookmark182) is responsible for filtering network traffic based on firewall rules to protect the system from malicious messages. To this end, the [Firewall](#_bookmark182) parses the firewall rules from the Manifest and configures the underlying firewall engine accordingly. The firewall engine can be realized in different ways (e.g. on the level of the TCP/IP stack or even closer to the hardware), which is considered to be an implementation detail.  Additionally, the [Firewall](#_bookmark182) supports handling of different modes (e.g. driving, parking, diagnostic session) by enabling/disabling firewall rules based on the active mode. For blocked messages security events to the [Adaptive Intrusion Detection System Manager](#_bookmark178) will be reported to support the AUTOSAR intrusion detection system. |

* + - 1. **Defined interfaces**



Adaptive Application



«use»

«aapRequiredPort»

«aapFunctionalCluster»

Firewall

daemon-based

+ SwitchFirewallState()

«aapAPI,aapPortInterface» FirewallStateSwitchInterface

**Figure 9.65: Interfaces of Firewall**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | FirewallStateSwitchInterface | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | FirewallStateSwitchInterface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Provides functionality to switch the firewall state. | |
| ***Operations:*** | SwitchFirewallState | This method triggers a switch of the firewall state. |

* + - 1. **Provided interfaces**

The [Firewall](#_bookmark182) does not provide interfaces to other Functional Clusters.

* + - 1. **Required interfaces**



«aapFunctionalCluster»

Firewall

daemon-based

«use» «use»



«aapFunctionalCluster»

Adaptive Intrusion Detection System Manager daemon-based

+ ReportEvent()

«aapPortInterface,aapAPI» EventReporter



Operating System

**Figure 9.66: Interfaces required by Firewall**

«aapInternal» TCP/IP Stack

|  |  |
| --- | --- |
| ***Interface*** | ***Purpose*** |
| TCP/IP Stack | The [Firewall](#_bookmark182) uses this interface to enable/disable firewall rules. |
| [Adaptive Intrusion Detection System Manager](#_bookmark178):: [EventReporter](#_bookmark179) | The [Firewall](#_bookmark182) uses this interface to report security events. |

**Table 9.22: Interfaces required by Firewall**

## Safety



«aapFunctionalCluster»

Platform Health Management

daemon-based

**Figure 9.67: Functional Clusters in category Safety**

### Platform Health Management

|  |  |
| --- | --- |
| ***Name:*** | Platform Health Management |
| ***Short Name:*** | phm |
| ***Category:*** | Safety |
| ***Daemon-based:*** | Yes |
| ***Responsibilities:*** | [Platform Health Management](#_bookmark186) performs (aliveness, logical, and deadline) supervision of Processes in safety-critical setups and reports failures to [State Management](#_bookmark76). [Platform](#_bookmark186) [Health Management](#_bookmark186) also controls the Watchdog that in turn supervises the [Platform Health](#_bookmark186) [Management](#_bookmark186).  An Alive Supervision checks that a supervised entity is not running too frequently and not too rarely. A Deadline Supervision checks that steps in a supervised entity are executed within the configured minimum and maximum time. A Logical Supervision checks that the control flow during execution matches the designed control flow. All types of supervision can be used independently and are performed based on reporting of Checkpoints by the supervised entity.  [State Management](#_bookmark76) and [Execution Management](#_bookmark61) are the fundamental Functional Clusters of the AUTOSAR Adaptive Platform and need to run and work properly in any case. Therefore, [Platform Health Management](#_bookmark186) shall always supervise the corresponding Processes for [State Management](#_bookmark76) and [Execution Management](#_bookmark61). Supervision failures in these Processes shall be recovered by a reset of the Machine because the normal way of error recovery (via [State Management](#_bookmark76) and [Execution Management](#_bookmark61)) is no longer reliable. |

* + - 1. **Defined interfaces**

The interfaces of [Platform Health Management](#_bookmark186) are categorized into interfaces for supervision (see Section [9.6.1.1.1](#_bookmark187)), interfaces for performing recovery actions (see Section [9.6.1.1.2](#_bookmark190)), and interfaces for hardware watchdog handling (see Section [9.6.1.1.3](#_bookmark195)).

* + - * 1. **Interfaces for supervision**

Processes that are supervised by [Platform Health Management](#_bookmark186) shall report via the [SupervisedEntity](#_bookmark189) interface when they have reached a certain checkpoint in their control flow (see Figure [9.68](#_bookmark188)). [Platform Health Management](#_bookmark186) independently mon- itors that all checkpoints configured in the Manifest have been reached in time and in the expected order (depending on the type of supervision).



Adaptive Application

«use»

«aapProvidedPort»



«aapFunctionalCluster»

Platform Health Management daemon-based

+ ReportCheckpoint()

«aapAPI,aapPortInterface» SupervisedEntity

**Figure 9.68:** **Interfaces for supervision**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | SupervisedEntity | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | PhmSupervisedEntityInterface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides functions to report checkpoints to [Platform Health Management](#_bookmark186). | |
| ***Operations:*** | ReportCheckpoint | Reports an occurrence of a checkpoint. |

* + - * 1. **Interfaces for recovery**

[Platform Health Management](#_bookmark186) defines the [RecoveryAction](#_bookmark192) API to trigger a recov- ery action in case a supervision failed (see Figure [9.69](#_bookmark191)).



«aapFunctionalCluster»

State Management daemon-based

«aapProvidedPort»

+ GetGlobalSupervisionStatus()

+ Offer()

+ StopOffer()

«aapCallbackMethod»

+ RecoveryHandler()

«aapPortInterface,aapAPI» Platform Health Management::

RecoveryAction

«use»



«aapFunctionalCluster»

Platform Health Management daemon-based

**Figure 9.69:** **Interfaces for recovery**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | RecoveryAction | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | PhmRecoveryActionInterface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides functions to control triggering of recovery actions, to determine the status of the supervision and a callback to perform recovery. | |
| ***Operations:*** | GetGlobalSupervisionStatus | Returns the status of global supervision that the supervised entity belongs to. |
| Offer | Enables potential invocations of the callback [RecoveryHandler()](#_bookmark194). |
| *RecoveryHandler* | Callback to be invoked by [Platform Health](#_bookmark186) [Management](#_bookmark186) upon a supervision failure. The handler invocation needs to be enabled before using [Offer()](#_bookmark193). |
| StopOffer | Disables potential invocations of the callback [RecoveryHandler()](#_bookmark194). |

* + - * 1. **Interfaces for watchdog handling**

[Platform Health Management](#_bookmark186) defines the [WatchdogInterface](#_bookmark197) extension API to in- teract with the hardware watchdog (see Figure [9.70](#_bookmark196)).



+ AliveNotification()

+ FireWatchdogReaction()

«aapNativeInterface,aapPlatformExtension» Platform Health Management::WatchdogInterface

Watchdog

«use»



«aapFunctionalCluster»

Platform Health Management daemon-based

**Figure 9.70:** **Interfaces for watchdog handling**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | WatchdogInterface | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Platform extension | |
| ***Description:*** | This interface provides functions to control the hardware watchdog. | |
| ***Operations:*** | AliveNotification | Called cyclically by [Platform Health Management](#_bookmark186) in configurable cycle time. Note: This time might differ from the cycle time of triggering the "real" hardware watchdog. If [Platform Health Management](#_bookmark186) does not report aliveness in configured time, [WatchdogInterface](#_bookmark197) shall initiate watchdog reaction. |
| FireWatchdogReaction | Initiates an error reaction of the hardware watchdog. |

* + - 1. **Provided interfaces**



«aapFunctionalClust...

Time Synchronization daemon-based



«aapFunctionalCluster» Update and Configuration Management

daemon-based



«aapFunctionalClust...

Cryptography daemon-based



«aapFunctionalClust...

State Management daemon-based



«aapFunctionalClust...

Execution Management daemon-based



«aapFunctionalClust... Diagnostic Management daemon-based



«use»

«use»

«use»

«use»

«use»

«use»

«aapRequiredPort» «aapRequiredPort» aapRequiredPort» «aapRequiredPort» «aapRequiredPort»«aapRequiredPort»

daemon-based

«aapFunctionalCluster» Platform Health Management

+ ReportCheckpoint()

«aapAPI,aapPortInterface» SupervisedEntity

**Figure 9.71: Users of the SupervisedEntity interface**

«

|  |  |
| --- | --- |
| ***Interface*** | ***Requiring functional clusters*** |
| [Platform Health Management](#_bookmark186)::[SupervisedEntity](#_bookmark189) | [Cryptography](#_bookmark150) |
| [Diagnostic Management](#_bookmark214) |
| [Execution Management](#_bookmark61) |
| [State Management](#_bookmark76) |
| [Time Synchronization](#_bookmark115) |
| [Update and Configuration Management](#_bookmark200) |

**Table 9.23: Interfaces provided by Platform Health Management to other Functional Clus- ters**

* + - 1. **Required interfaces**



«aapFunctionalCluster»

Platform Health Management

daemon-based

«use» «use» «use»

+ GetGlobalSupervisionStatus()

+ Offer()

+ StopOffer()

«aapCallbackMethod»

+ RecoveryHandler()

«aapPortInterface,aapAPI» Platform Health Management:: RecoveryAction

«aapProvidedPort»



Watchdog

+ AliveNotification()

+ FireWatchdogReaction()

«aapNativeInterface,aapPlatformExtens...

Platform Health Management:: WatchdogInterface



«aapFunctionalCluster»

Execution Management daemon-based

+ ReportExecutionState(ExecutionState): Result

«aapAPI,aapNativeInterface» Execution Management::ExecutionClient



«aapFunctionalCluster»

State Management daemon-based

**Figure 9.72: Interfaces required by Platform Health Management**

|  |  |
| --- | --- |
| ***Interface*** | ***Purpose*** |
| [Execution Management](#_bookmark61)::[ExecutionClient](#_bookmark64) | [Platform Health Management](#_bookmark186) uses this interface to report the state of its daemon process to [Execution Management](#_bookmark61). |
| [Log and Trace](#_bookmark90)::[Logger](#_bookmark91) | [Platform Health Management](#_bookmark186) shall use this interface to log standardized messages. |
| [Platform Health Management](#_bookmark186)::[RecoveryAction](#_bookmark192) | [Platform Health Management](#_bookmark186) uses this interface to trigger failure recovery. |
| [Platform Health Management](#_bookmark186)::[Watchdog](#_bookmark197) [Interface](#_bookmark197) | [Platform Health Management](#_bookmark186) uses this interface to control the hardware watchdog. |
| [Registry](#_bookmark210)::[ManifestAccessor](#_bookmark211) | [Platform Health Management](#_bookmark186) shall use this interface to read information about [SupervisedEntities](#_bookmark189) from the Manifests. |

**Table 9.24: Interfaces required by Platform Health Management**

## Configuration



«aapFunctionalCluster» Update and Configuration Management

daemon-based



«aapFunctionalCluster» Registry

**Figure 9.73: Functional Clusters in category Configuration**

### Update and Configuration Management

|  |  |
| --- | --- |
| ***Name:*** | Update and Configuration Management |
| ***Short Name:*** | ucm |
| ***Category:*** | Configuration |
| ***Daemon-based:*** | Yes |
| ***Responsibilities:*** | [Update and Configuration Management](#_bookmark200) is responsible for updating, installing, removing and keeping a record of the software on an AUTOSAR Adaptive Platform in a safe and secure way.  Hereby, [Update and Configuration Management](#_bookmark200) enables to update the software and its configuration flexibly through over-the-air updates (OTA).  [Update and Configuration Management](#_bookmark200) is separated into two main components UCM Master and UCM Subordinate. UCM Subordinate controls the update process on the local Adaptive Platform. UCM Master controls an update of the software in the entire vehicle. |

* + - 1. **Defined interfaces**

The interfaces of [Update and Configuration Management](#_bookmark200) are categorized into interfaces for UCM Subordinate (see Section [9.7.1.1.1](#_bookmark201)), interfaces for UCM Master (see Section [9.7.1.1.2](#_bookmark203)), and interfaces for the D-PDU API (see Section [9.7.1.1.3](#_bookmark204)).

* + - * 1. **UCM Subordinate**



Adaptive Application

Diagnostic Application

«use»

«aapRequiredPort»

«use»

«aapRequiredPort»



«aapAccessControlled, aapServiceMethod»

+ Activate()

+ Cancel()

+ DeleteTransfer()

+ Finish()

+ GetHistory()

+ GetId()

+ GetSwClusterChangeInfo()

+ GetSwClusterDescription()

+ GetSwClusterInfo()

+ GetSwPackages()

+ GetSwProcessProgress()

+ ProcessSwPackage()

+ RevertProcessedSwPackages()

+ Rollback()

+ TransferData()

+ TransferExit()

+ TransferStart()

«aapAccessControlled, aapServiceField»

+ CurrentStatus

«aapAPI,aapAraComServiceInterface» PackageManagement

«aapProvidedPort»

|  |  |  |  |
| --- | --- | --- | --- |
| «aapFunction Update and Configura | | alCluster»  tion Management | |
|  | UCM Subordinate | |  |
| daemon-based | | | |

**Figure 9.74: Interfaces of UCM Subordinate**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | PackageManagement | |
| ***Technology:*** | ara::com service interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides functionality for managing and transferring Software Packages to an UCM Subordinate. | |
| ***Operations:*** | Activate | Activates the processed components. |
| Cancel | Aborts an ongoing processing of a Software Package. |
| DeleteTransfer | Delete a transferred Software Package. |
| Finish | Finishes the processing for the current set of processed Software Packages. It does a cleanup of all data of the processing including the sources of the Software Packages. |
| GetHistory | Retrieve all actions that have been performed by UCM Subordinate. |
| GetId | Get the UCM Subordinate Instance Identifier. |

*Δ*

|  |  |  |
| --- | --- | --- |
|  | GetSwClusterChangeInfo | Get a list of pending changes to the set of Software Clusters on the Adaptive Platform. The returned list includes all Software Clusters that are to be added, updated or removed. The list of changes is extended in the course of processing Software Packages. |
| GetSwClusterDescription | Get the general information of the Software Clusters present in the platform. |
| GetSwClusterInfo | Get a list of Software Clusters that are in state k Present. |
| GetSwPackages | Get the Software Packages that available in UCM Subordinate. |
| GetSwProcessProgress | Get the progress of the currently processed Software Package. |
| ProcessSwPackage | Process a previously transferred Software Package. |
| RevertProcessedSwPackages | Revert the changes done by processing (by calling [ProcessSwPackage()](#_bookmark202)) of one or several Software Packages. |
| Rollback | Rollback the system to the state before the packages were processed. |
| TransferData | Block-wise transfer of a Software Package to UCM Subordinate. |
| TransferExit | Finish the transfer of a Software Package to UCM Subordinate. |
| TransferStart | Start the transfer of a Software Package. |
| ***Fields:*** | CurrentStatus | The current status of the UCM Subordinate. |

* + - * 1. **UCM Master**

|  |  |  |  |
| --- | --- | --- | --- |
| Adaptive Application  OTA Client  Vehicle Driver Interface  Vehicle State Manager | | | |
|  |  |  |  |

«aapProvidedPort»



«use»

«aapRequiredPort»

«aapAPI,aapAraComServiceInterface» VehiclePackageManagement

«aapAccessControlled, aapServiceField»

+ RequestedPackage

+ SafetyState

+ TransferState

«aapAccessControlled, aapServiceMeth...

+ AllowCampaign()

+ CancelCampaign()

+ DeleteTransfer()

+ GetCampaignHistory()

+ GetSwClusterInfo()

+ GetSwPackages()

+ SwPackageInventory()

+ TransferData()

+ TransferExit()

+ TransferStart()

+ TransferVehiclePackage()



«use»

«aapRequiredPort»

«aapAccessControlled, aapServiceMeth...

+ AllowCampaign()

+ CancelCampaign()

+ DriverApproval()

+ GetCampaignHistory()

+ GetSwClusterDescription()

+ GetSwPackageDescription()

+ GetSwProcessProgress()

+ GetSwTransferProgress()

«aapAccessControlled, aapServiceField»

+ ApprovalRequired

+ CampaignState

+ SafetyPolicy

+ SafetyState

«aapAPI,aapAraComServiceInterface» VehicleDriverApplication

«aapProvidedPort»

«aapProvidedPort»



«use»

«aapRequiredPort»

«aapAccessControlled, aapServiceMeth...

+ SafetyState()

«aapAccessControlled, aapServiceField»

+ SafetyPolicy

«aapAPI,aapAraComServiceInterface» VehicleStateManager



«aapFunctionalCluster»

Update and Configuration Management

daemon-based

UCM Master

**Figure 9.75: Interfaces of UCM Master**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | VehiclePackageManagement | |
| ***Technology:*** | ara::com service interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides functionality for managing and transferring Vehicle Packages and  Software Packages to UCM Master. | |
| ***Operations:*** | AllowCampaign | Allows a new campaign to start. |
| CancelCampaign | Aborts an ongoing campaign processing of a Vehicle Package. |
| DeleteTransfer | Delete a transferred Software Package or Vehicle Package. |
| GetCampaignHistory | Retrieve all actions that have been performed by UCM Master. |
| GetSwClusterInfo | Get a list of SoftwareClusters that are in state k Present. |
| GetSwPackages | Get the Software Packages that are part of current campaign handled by UCM Master. |
| SwPackageInventory | Performs an inventory of all Software Packages. |
| TransferData | Block-wise transfer of a Software Package or  Vehicle Package to UCM Master. |

*Δ*

|  |  |  |
| --- | --- | --- |
|  | TransferExit | Finish the transfer of a Software Package or  Vehicle Package to UCM Master. |
| TransferStart | Start the transfer of a Software Package to UCM Master. |
| TransferVehiclePackage | Start the transfer of a Vehicle Package to UCM Master. |
| ***Fields:*** | RequestedPackage | Software Package to be transferred to UCM Master. |
| SafetyState | Vehicle state computed by the Vehicle State Manager Adaptive Application. |
| TransferState | The current status of a campaign from an OTA Client  perspective. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | VehicleDriverApplication | |
| ***Technology:*** | ara::com service interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides functionality to interact with the vehicle driver, for example to approve updates. | |
| ***Operations:*** | AllowCampaign | Allow a new campaign to start. |
| CancelCampaign | Aborts an ongoing campaign processing of a Vehicle Package. |
| DriverApproval | Inform UCM Master of the driver’s notification resolution (approve or reject). |
| GetCampaignHistory | Retrieve all actions that have been performed by UCM Master. |
| GetSwClusterDescription | Get the general information of the Software Clusters present in the Adaptive Platform. |
| GetSwPackageDescription | Get the general information of the Software Packages that are part of current campaign handled by UCM Master. |
| GetSwProcessProgress | Get the progress of the current package processing. |
| GetSwTransferProgress | Get the progress of the current package transfer. |
| ***Fields:*** | ApprovalRequired | Flag to inform an Adaptive Application if approval from a driver is required at current state based on the Vehicle Package Manifest. |
| CampaignState | The current status of campaign. |
| SafetyPolicy | Safety policy from the Vehicle Package to be computed by the Vehicle State Manager Adaptive Application. |
| SafetyState | Vehicle state computed by the Vehicle State Manager Adaptive Application. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | VehicleStateManager | |
| ***Technology:*** | ara::com service interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides functionality for a Vehicle State Manager Adaptive Application to inform  UCM Master about the safety state and policy of the vehicle. | |
| ***Operations:*** | SafetyState | Called by the Vehicle State Manager Adaptive Application when safety state is changed. |
| ***Fields:*** | SafetyPolicy | Safety policy from the Vehicle Package to be computed by the Vehicle State Manager Adaptive Application. |

* + - * 1. **D-PDU API**



Adaptive Application

Flashing Adapter

«use»



«aapAPI,aapNativeInterface» D-PDU API

«aapFunctionalCluster» Update and Configuration Management

D-PDU API

daemon-based

**Figure 9.76: Interfaces of UCM Master**

|  |  |
| --- | --- |
| ***Name:*** | D-PDU API |
| ***Technology:*** | Native interface |
| ***Usage:*** | Public API |
| ***Description:*** | This interface represents the Diagnostic Protocol Data Unit Application Programming Interface as specified in ISO 22900-2 . This interface is not detailed in this document. |

* + - 1. **Provided interfaces**

[Update and Configuration Management](#_bookmark200) does not provide any interfaces to other Functional Clusters.

* + - 1. **Required interfaces**

Figures [9.77](#_bookmark205) and [9.78](#_bookmark206) show the interfaces that are required by [Update and Con-](#_bookmark200) [figuration Management](#_bookmark200). These interface are thus required by both UCM Subor- dinate and UCM Master.



«aapFunctionalCluster»

Update and Configuration Management

daemon-based

«use»

«aapRequiredPort»

«use»

«aapRequiredPort»

«use»

«aapRequiredPort»



«aapAPI,aapPortInterface» SynchronizedTimeBaseConsumer

+ GetCurrentTime()

+ GetRateDeviation()

+ GetTimeWithStatus(): SynchronizedTimeBaseStatus

+ RegisterStatusChangeNotifier()

+ RegisterSynchronizationStateChangeNotifier()

+ RegisterTimeLeapNotifier()

+ RegisterTimePrecisionMeasurementNotifier()

+ RegisterTimeValidationNotification()

+ UnregisterStatusChangeNotifier()

+ UnregisterSynchronizationStateChangeNotifier()

+ UnregisterTimeLeapNotifier()

+ UnregisterTimePrecisionMeasurementNotifier()

+ UnregisterTimeValidationNotification()

«aapFunctionalCluster» Time Synchronization

daemon-based



«aapAPI,aapPortInterface» FileStorage

«aapAPI,aapPortInterface» KeyValueStorage

+ DeleteFile()

+ FileExists()

+ GetAllFileNames()

+ GetCurrentFileSize()

+ GetFileInfo()

+ OpenFileReadOnly(): ReadAccessor

+ OpenFileReadWrite(): ReadWriteAccessor

+ OpenFileWriteOnly(): ReadWriteAccessor

+ RecoverFile()

+ ResetFile()

+ DiscardPendingChanges()

+ GetAllKeys()

+ GetValue()

+ KeyExists()

+ RecoverKey()

+ RemoveAllKeys()

+ RemoveKey()

+ ResetKey()

+ SetValue()

+ SyncToStorage()

«aapFunctionalCluster» Persistency

**Figure 9.77:** **Interfaces required by Update and Configuration Management (part 1)**



daemon-based

«aapFunctionalCluster»

Update and Configuration Management

«use» «use»



«aapAPI,aapNativeInterface» Logger

+ IsEnabled()

+ Log(MsgId, Params)

+ LogDebug(): LogStream

+ LogError(): LogStream

+ LogFatal(): LogStream

+ LogInfo(): LogStream

+ LogVerbose(): LogStream

+ LogWarn(): LogStream

+ WithLevel(): LogStream

«aapFunctionalCluster» Log and Trace



«aapAPI,aapNativeInterface» ExecutionClient

+ ReportExecutionState(ExecutionState): Result

«aapFunctionalCluster»

Execution Management daemon-based

**Figure 9.78:** **Interfaces required by Update and Configuration Management (part 2)**

Figure [9.79](#_bookmark207) shows the interfaces that are only required by UCM Subordinate. Cor- respondingly, Figure [9.80](#_bookmark208) shows the interfaces that are only required by UCM Master.

«aapFunctionalCluster»



Update and Configuration Management

UCM Subordinate

daemon-based

«use»

«aapRequiredPort»

«aapAccessControlled, aapService...

+ PrepareRollback()

+ PrepareUpdate()

+ RequestUpdateSession()

+ ResetMachine()

+ StopUpdateSession()

+ VerifyUpdate()

«aapAraComServiceInterface,aapIn...

UpdateRequest

«aapProvidedPort»

«use»

«aapAPI,aapNativeInterf...

CryptoStack



«aapFunctionalCluster»

State Management daemon-based

**Figure 9.79:** **Interfaces required by UCM Subordinate**



«aapFunctionalClust...

Cryptography daemon-based



«aapFunctionalCluster»

Update and Configuration Management

daemon-based

«use»

«aapRequiredPort»

«aapAPI,aapAraComServiceInterface» PackageManagement

«aapProvidedPort»

«aapAccessControlled, aapServiceField»

+ CurrentStatus

«aapAccessControlled, aapServiceMeth...

+ Activate()

+ Cancel()

+ DeleteTransfer()

+ Finish()

+ GetHistory()

+ GetId()

+ GetSwClusterChangeInfo()

+ GetSwClusterDescription()

+ GetSwClusterInfo()

+ GetSwPackages()

+ GetSwProcessProgress()

+ ProcessSwPackage()

+ RevertProcessedSwPackages()

+ Rollback()

+ TransferData()

+ TransferExit()

+ TransferStart()

UCM Subordinate

UCM Master

«aapProvidedPort»



Adaptive Application

Flashing Adapter

**Figure 9.80:** **Interfaces required by UCM Master**

|  |  |
| --- | --- |
| ***Interface*** | ***Purpose*** |
| [Execution Management](#_bookmark61)::[ExecutionClient](#_bookmark64) | This interface shall be used by the daemon process(es) inside [Update](#_bookmark200) [and Configuration Management](#_bookmark200) to report their execution state to [Execution Management](#_bookmark61). |
| [Log and Trace](#_bookmark90)::[Logger](#_bookmark91) | [Update and Configuration Management](#_bookmark200) shall use this interface to log standardized messages. |
| [Persistency](#_bookmark124)::[FileStorage](#_bookmark130) | This interface should be used to store files, for example downloaded packages. |
| [Persistency](#_bookmark124)::[KeyValueStorage](#_bookmark141) | This interface should be used to store internal state of [Update and](#_bookmark200) [Configuration Management](#_bookmark200). |
| [Persistency](#_bookmark124)::[ReadAccessor](#_bookmark134) | This interface should be used to store files, for example downloaded packages. |
| [Persistency](#_bookmark124)::[ReadWriteAccessor](#_bookmark135) | This interface should be used to store files, for example downloaded packages. |
| [Platform Health Management](#_bookmark186)::[SupervisedEntity](#_bookmark189) | This interface should be used to supervise the daemon process(es) of  [Update and Configuration Management](#_bookmark200). |
| [Registry](#_bookmark210)::[ManifestAccessor](#_bookmark211) | [Update and Configuration Management](#_bookmark200) shall use this interface to read information about its configuration from the Manifests. |
| [Time Synchronization](#_bookmark115)::[SynchronizedTimeBase](#_bookmark120) [Consumer](#_bookmark120) | [Update and Configuration Management](#_bookmark200) shall use this interface to get latest timestamp. |

**Table 9.25: Interfaces required by Update and Configuration Management**

### Registry

|  |  |
| --- | --- |
| ***Name:*** | Registry |
| ***Short Name:*** | n/a |
| ***Category:*** | Configuration |
| ***Daemon-based:*** | No |
| ***Responsibilities:*** | The [Registry](#_bookmark210) is an internal component of the AUTOSAR Adaptive Platform that provides access the information stored in Manifests. It is not intended to be used by [Adaptive Application](#_bookmark24)s directly. |

* + - 1. **Defined interfaces**



«aapFunctionalClust... Diagnostic Management

daemon-based



«aapFunctionalClust...

Execution Management daemon-based



«aapFunctionalClust...

Platform Health Management

daemon-based



«aapFunctionalClust...

Communication Management



«aapFunctionalClust... Network Management



«aapFunctionalClust...

Time Synchronization daemon-based



«aapFunctionalCluster» Update and Configuration Management

daemon-based

«use»



«aapFunctionalClust...

Cryptography daemon-based

«use»

«use»

«use»

«use»

«use»

«use»

«use»

«aapInternal,aapNativeInterface» ManifestAccessor

«use» «use» «use»



«aapFunctionalClust...

Persistency



«aapFunctionalClust... Identity and Access Management



«aapFunctionalClust...

Log and Trace



«aapFunctionalCluster» Registry

**Figure 9.81: Interfaces of Registry**

|  |  |
| --- | --- |
| ***Name:*** | ManifestAccessor |
| ***Technology:*** | Native interface |
| ***Usage:*** | Internal |
| ***Description:*** | This interface provides functionality to read information that was modeled in the Manifest(s). This interface is not detailed in this document. |

* + - 1. **Provided interfaces**



«aapFunctionalClust... Diagnostic Management

daemon-based



«aapFunctionalClust...

Execution Management daemon-based



«aapFunctionalClust...

Platform Health Management

daemon-based



«aapFunctionalClust...

Communication Management



«aapFunctionalClust... Network Management



«aapFunctionalClust...

Time Synchronization daemon-based



«aapFunctionalCluster» Update and Configuration Management

daemon-based

«use»



«aapFunctionalClust...

Cryptography daemon-based

«use»

«use»

«use»

«use»

«use»

«use»

«use»

«aapInternal,aapNativeInterface» ManifestAccessor

«use» «use» «use»



«aapFunctionalClust...

Persistency



«aapFunctionalClust... Identity and Access Management



«aapFunctionalClust...

Log and Trace



«aapFunctionalCluster» Registry

**Figure 9.82: Users of the Registry interfaces**

|  |  |
| --- | --- |
| ***Interface*** | ***Requiring functional clusters*** |
| [Registry](#_bookmark210)::[ManifestAccessor](#_bookmark211) | [Communication Management](#_bookmark105) |
| [Cryptography](#_bookmark150) |
| [Diagnostic Management](#_bookmark214) |
| [Execution Management](#_bookmark61) |
| [Identity and Access Management](#_bookmark175) |
| [Log and Trace](#_bookmark90) |
| [Network Management](#_bookmark112) |
| [Persistency](#_bookmark124) |
| [Platform Health Management](#_bookmark186) |
| [Time Synchronization](#_bookmark115) |
| [Update and Configuration Management](#_bookmark200) |

**Table 9.26: Interfaces provided by Registry to other Functional Clusters**

* + - 1. **Required interfaces**



«aapFunctionalCluster» Registry

«use»



Operating System

«aapInternal» Non-volatile Storage

**Figure 9.83: Interfaces required by Registry**

|  |  |
| --- | --- |
| ***Interface*** | ***Purpose*** |
| Non-volatile Storage | [Registry](#_bookmark210) shall use this interface to read the information from the  Manifest(s). |

**Table 9.27: Interfaces required by Registry**

## Diagnostics



«aapFunctionalCluster»

Diagnostic Management

daemon-based

**Figure 9.84: Functional Clusters in category Diagnostics**

### Diagnostic Management

|  |  |
| --- | --- |
| ***Name:*** | Diagnostic Management |
| ***Short Name:*** | diag |
| ***Category:*** | Diagnostics |
| ***Daemon-based:*** | Yes |
| ***Responsibilities:*** | [Diagnostic Management](#_bookmark214) is responsible for handling diagnostic events produced by the individual Processes running in an AUTOSAR Runtime for Adaptive Applications. [Diagnostic](#_bookmark214) [Management](#_bookmark214) stores such events and the associated data persistently according to rendition policies. [Diagnostic Management](#_bookmark214) also provides access to diagnostic data for external Diagnostic Clients via standardized network protocols (ISO 14229-5 (UDSonIP) which is based on the ISO 14229-1 (UDS) and ISO 13400-2 (DoIP)). |

* + - 1. **Defined interfaces**
         1. **Common interfaces**

«use»

«use»

«aapRequiredPort»



Adaptive Application

«aapNativeInterface,aapAPI» Conversation

«aapAPI,aapPortInterface» DTCInformation

+ GetActivityStatus()

+ GetAllConversations()

+ GetConversation(): Conversation

+ GetConversationIdentifier()

+ GetCurrentActiveConversations()

+ GetDiagnosticSecurityLevel()

+ GetDiagnosticSecurityLevelShortName()

+ GetDiagnosticSession()

+ GetDiagnosticSessionShortName()

+ ResetToDefaultSession()

+ SetActivityNotifier()

+ SetDiagnosticSessionNotifier()

+ SetSecurityLevelNotifier()

+ Clear()

+ EnableControlDtc()

+ GetControlDTCStatus()

+ GetCurrentStatus()

+ GetEventMemoryOverflow()

+ GetNumberOfStoredEntries()

+ SetControlDtcStatusNotifier()

+ SetDTCStatusChangedNotifier()

+ SetEventMemoryOverflowNotifier()

+ SetNumberOfStoredEntriesNotifier()

+ SetSnapshotRecordUpdatedNotifier()

«aapFunctionalCluster»

Diagnostic Management

daemon-based

**Figure 9.85: Common interfaces of Diagnostic Management (1 of 2)**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | Conversation | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides functionality to handle diagnostic conversations. | |
| ***Operations:*** | GetActivityStatus | Represents the status of an active conversation. |
| GetAllConversations | Get all possible conversations. |
| GetConversation | Get one conversation based on given meta information. |

*q*

*Δ*

|  |  |  |
| --- | --- | --- |
|  | GetConversationIdentifier | Getter for the current identification properties of the active conversation. |
| GetCurrentActiveConversations | Get all currently active conversations. |
| GetDiagnosticSecurityLevel | Represents the current active diagnostic Security Level of an active conversation. |
| GetDiagnosticSecurityLevelShortName | Converts the given diagnostic SecurityLevel into the  ShortName. |
| GetDiagnosticSession | Represents the current active diagnostic session of an active conversation. |
| GetDiagnosticSessionShortName | Converts the given diagnostic session into the Short Name. |
| ResetToDefaultSession | Method to reset the current session to the default session. |
| SetActivityNotifier | Register a notifier function which is called if the activity is changed. |
| SetDiagnosticSessionNotifier | Register a notifier function which is called if the Session is changed. |
| SetSecurityLevelNotifier | Register a notifier function which is called if the  SecurityLevel is changed. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | DTCInformation | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | DiagnosticDTCInformationInterface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides operations on DTC information per configured DiagnosticMemory  destination. | |
| ***Operations:*** | Clear | Method for Clearing a DTC or a group of DTCs. |
| EnableControlDtc | Enforce restoring ControlDTCStatus setting to enabled in case the monitor has some conditions or states demands to do so. |
| GetControlDTCStatus | Contains the current status of the ControlDTCStatus. |
| GetCurrentStatus | Retrieves the current UDS DTC status byte of the given DTC identifier. |
| GetEventMemoryOverflow | Contains the current event memory overflow status. |
| GetNumberOfStoredEntries | Contains the number of currently stored fault memory entries. |
| SetControlDtcStatusNotifier | Registers a notifier function which is called if the control DTC setting is changed. |
| SetDTCStatusChangedNotifier | Register a notifier function which is called if a UDS DTC status is changed. |
| SetEventMemoryOverflowNotifier | Register a notifier function which is called if the current event memory overflow status changed. |
| SetNumberOfStoredEntriesNotifier | Register a notifier function which is called if the number of currently stored fault memory entries changed. |
| SetSnapshotRecordUpdatedNotifier | Register a notifier function which is called if the SnapshotRecord is changed. |



Adaptive Application

«use» «use»



daemon-based

«aapFunctionalCluster» Diagnostic Management

+ GetContext()

+ GetValue()

+ IsCanceled()

+ SetNotifier()

«aapAPI,aapNativeInterface» MetaInfo

«aapAPI,aapNativeInterface» CancellationHandler

**Figure 9.86: Common interfaces of Diagnostic Management (2 of 2)**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | CancellationHandler | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface holds a shared state if the processing should be canceled. | |
| ***Operations:*** | IsCanceled | Returns true in if the diagnostic service execution is canceled. |
| SetNotifier | Registers a notifier function which is called if the diagnostic service execution is canceled. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | MetaInfo | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface specifies a mechanism to provide meta information, i.e. from transport protocol layer, to an interested application. | |
| ***Operations:*** | GetContext | Get the context of the invocation. |
| GetValue | Get the metainfo value for a given key. |

* + - * 1. **Interfaces for request handling**



Adaptive Application

«use»

«aapRequiredPort»

«use»

«aapRequiredPort»

«use»

«aapRequiredPort»



«aapAPI,aapPortInterface» «aapAPI,aapPortInterface» «aapAPI,aapPortInterface» Condition OperationCycle Indicator

+ GetCondition()

+ SetCondition()

+ GetOperationCycle()

+ SetNotifier()

+ SetOperationCycle()

+ GetIndicator()

+ SetNotifier()

«aapFunctionalCluster» Diagnostic Management

daemon-based

|  |  |  |
| --- | --- | --- |
| ***Name:*** | Condition | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | DiagnosticConditionInterface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides functionality for condition management. | |
| ***Operations:*** | GetCondition | Get the current condition. |
| SetCondition | Set the current condition. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | OperationCycle | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | DiagnosticOperationCycleInterface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides functionality for handling of operation cycles. | |
| ***Operations:*** | GetOperationCycle | Get the current OperationCycle. |
| SetNotifier | Registers a notifier function which is called if the  OperationCycle is changed. |
| SetOperationCycle | Set the current OperationCycle. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | Indicator | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | DiagnosticIndicatorInterface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides functionality for handling indicators. | |
| ***Operations:*** | GetIndicator | Get current Indicator. |
| SetNotifier | Register a notifier function which is called if the indicator is updated. |

Adaptive Application

+ CompareKey()

+ GetSeed()

+ Offer()

+ StopOffer()

«aapPortInterface,aapAPI» SecurityAccess

+ Confirmation()

+ Offer()

+ StopOffer()

+ Validate()

«aapPortInterface,aapAPI» ServiceValidation

«use»



«aapRequiredPort»

«use»

«aapRequiredPort»



«aapFunctionalCluster»

Diagnostic Management

daemon-based

|  |  |  |
| --- | --- | --- |
| ***Name:*** | SecurityAccess | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | DiagnosticSecurityLevelInterface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides functionality for handling SecurityAccess requests. | |
| ***Operations:*** | CompareKey | This method is called, when a diagnostic request has been finished, to notify about the outcome. |
| GetSeed | Called for any request message. |
| Offer | Enable forwarding of request messages from  [Diagnostic Management](#_bookmark214). |
| StopOffer | Disable forwarding of request messages from  [Diagnostic Management](#_bookmark214). |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | ServiceValidation | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | DiagnosticServiceValidationInterface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides functionality for handling ServiceValidation requests. | |
| ***Operations:*** | Confirmation | This method is called, when a diagnostic request has been finished, to notify about the outcome. |
| Offer | Enable forwarding of request messages from  [Diagnostic Management](#_bookmark214). |
| StopOffer | Disable forwarding of request messages from  [Diagnostic Management](#_bookmark214). |
| Validate | Called for any request message. |



Adaptive Application

«use»

«aapRequiredPort»

«use»

«aapRequiredPort»

«aapPortInterface,a...

Monitor

«aapPortInterface,aapAPI» Event

«aapPortInterface,aap... GenericDataIdentifier

«aapPortInterface,aap... GenericUDSService

«aapPortInterface,aap...

GenericRoutine

+ Offer()

+ ReportMonitorAction()

+ StopOffer()

+ GetDebouncingStatus()

+ GetDTCNumber()

+ GetEventStatus()

+ GetFaultDetectionCounter()

+ GetLatchedWIRStatus()

+ GetTestComplete()

+ SetEventStatusChangedNotifier()

+ SetLatchedWIRStatus()

+ Offer()

+ Read()

+ StopOffer()

+ Write()

+ HandleMessage()

+ Offer()

+ StopOffer()

+ Offer()

+ RequestResults()

+ Start()

+ Stop()

+ StopOffer()

«use»

«aapRequiredPort»

«aapFunctionalCluster» Diagnostic Management

«use»

«aapRequiredPort»

«use»

«aapRequiredPort»

daemon-based

**Figure 9.89: Interfaces for generic request handling**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | Monitor | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | DiagnosticMonitorInterface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides functionality to report qualified and unqualified test results and to control debouncing options. | |
| ***Operations:*** | Offer | Enable forwarding of request messages from  [Diagnostic Management](#_bookmark214). |
| ReportMonitorAction | Report the status information being relevant for error monitoring paths. |
| StopOffer | Disable forwarding of request messages from  [Diagnostic Management](#_bookmark214). |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | Event | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | DiagnosticEventInterface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface defines functionality for diagnostic events. | |
| ***Operations:*** | GetDTCNumber | Returns the DTC-ID related to this event instance. |
| GetDebouncingStatus | Get the current debouncing status. |
| GetEventStatus | Returns the current diagnostic event status. |
| GetFaultDetectionCounter | Returns the current value of Fault Detection Counter of this event. |
| GetLatchedWIRStatus | Returns the current warning indicator status. |
| GetTestComplete | Get the status if the event has matured to test completed (corresponds to FDC = -128 or FDC = 127). |
| SetEventStatusChangedNotifier | Register a notifier function which is called if a diagnostic event is changed. |
| SetLatchedWIRStatus | Set the warning indicator status. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | GenericDataIdentifier | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | DiagnosticDataIdentifierGenericInterface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Generic interface to handle ReadDataByIdentifier and WriteDataByIdentifier requests. | |
| ***Operations:*** | Offer | Enable forwarding of request messages from  [Diagnostic Management](#_bookmark214). |
| Read | Called for a ReadDataByIdentifier request for this  DiagnosticDataIdentifier. |
| StopOffer | Disable forwarding of request messages from  [Diagnostic Management](#_bookmark214). |
| Write | Called for a WriteDataByIdentifier request for this  DiagnosticDataIdentifier. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | GenericUDSService | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | DiagnosticGenericUdsInterface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Generic interface to handle UDS messages. | |
| ***Operations:*** | HandleMessage | Handles an UDS request message. |
| Offer | Enable forwarding of request messages from  [Diagnostic Management](#_bookmark214). |
| StopOffer | Disable forwarding of request messages from  [Diagnostic Management](#_bookmark214). |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | GenericRoutine | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | DiagnosticRoutineGenericInterface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Generic interface to handle RoutineControl requests. | |
| ***Operations:*** | Offer | Enable forwarding of request messages from  [Diagnostic Management](#_bookmark214). |
| RequestResults | Called for RoutineControl with SubFunction RequestResults request for this Diagnostic RoutineIdentifier. |
| Start | Called for RoutineControl with SubFunction Start request for this DiagnosticRoutine Identifier. |
| Stop | Called for RoutineControl with SubFunction Stop  request for this DiagnosticRoutineIdentifier. |
| StopOffer | Disable forwarding of request messages from  [Diagnostic Management](#_bookmark214). |

Adaptive Application



«aapPortInterface,aapAPI» Routine

«aapPortInterface,aapAPI» DataIdentifier

«aapPortInterface,aapAPI» DataElement

+ Offer()

+ RequestResults()

+ Start()

+ Stop()

+ StopOffer()

+ Offer()

+ Read()

+ StopOffer()

+ Write()

+ Offer()

+ Read()

+ StopOffer()

«use»

«aapRequiredPort»

«use»

«aapRequiredPort»

«use»

«aapRequiredPort»



daemon-based

«aapFunctionalCluster» Diagnostic Management

**Figure 9.90: Generated interfaces for request handling**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | DataIdentifier | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | Yes | |
| ***Meta-model interface type:*** | DiagnosticDataIdentifierInterface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Generated interface to handle ReadDataByIdentifier and WriteDataByIdentifier  requests. | |
| ***Operations:*** | Offer | Enable forwarding of request messages from  [Diagnostic Management](#_bookmark214). |
| Read | Called for a ReadDataByIdentifier request for this  DiagnosticDataIdentifier. |
| StopOffer | Disable forwarding of request messages from  Diagnostic Management. |
| Write | Called for a WriteDataByIdentifier request for this  DiagnosticDataIdentifier. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | DataElement | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | Yes | |
| ***Meta-model interface type:*** | DiagnosticDataElementInterface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Generated interface to handle read requests for DataElements. | |
| ***Operations:*** | Offer | Enable forwarding of request messages from  [Diagnostic Management](#_bookmark214). |
| Read | Called for reading a DataElement. |
| StopOffer | Disable forwarding of request messages from  [Diagnostic Management](#_bookmark214). |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | Routine | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | Yes | |
| ***Meta-model interface type:*** | DiagnosticRoutineInterface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Generated interface to handle RoutineControl requests. | |
| ***Operations:*** | Offer | Enable forwarding of request messages from  [Diagnostic Management](#_bookmark214). |
| RequestResults | Called for RoutineControl with SubFunction RequestResults request for this Diagnostic RoutineIdentifier. |
| Start | Called for RoutineControl with SubFunction Start request for this DiagnosticRoutine Identifier. |
| Stop | Called for RoutineControl with SubFunction Stop  request for this DiagnosticRoutineIdentifier. |
| StopOffer | Disable forwarding of request messages from  [Diagnostic Management](#_bookmark214). |

* + - * 1. **Interfaces for Upload and Download**



Adaptive Application

Diagnostic Application

«aapProvidedPort» «aapProvidedPort»



«aapPortInterface,aap...

DownloadService

+ DownloadData()

+ Offer()

+ RequestDownload()

+ RequestDownloadExit()

+ StopOffer()



«aapPortInterface,aap...

UploadService

+ Offer()

+ RequestUpload()

+ RequestUploadExit()

+ StopOffer()

+ UploadData()

«use» «use»



«aapFunctionalCluster» daemon-based Diagnostic Management

**Figure 9.91: Interfaces for Upload and Download**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | UploadService | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | DiagnosticUploadInterface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Upload service interface. | |
| ***Operations:*** | Offer | Enable forwarding of request messages from  [Diagnostic Management](#_bookmark214). |
| RequestUpload | Called for RequestUpload. |
| RequestUploadExit | Called for RequestTransferExit. |
| StopOffer | Disable forwarding of request messages from  [Diagnostic Management](#_bookmark214). |
| UploadData | Called for TransferData following a previous  RequestUpload. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | DownloadService | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | DiagnosticDownloadInterface | |
| ***Usage:*** | Public API | |
| ***Description:*** | Download service interface. | |
| ***Operations:*** | DownloadData | Called for TransferData following a previous  RequestDownload. |
| Offer | Enable forwarding of request messages from  [Diagnostic Management](#_bookmark214). |
| RequestDownload | Called for RequestDownload. |

*Δ*

|  |  |  |
| --- | --- | --- |
|  | RequestDownloadExit | Called for RequestTransferExit. |
| StopOffer | Disable forwarding of request messages from  [Diagnostic Management](#_bookmark214). |

* + - * 1. **Interfaces for State Management**

«aapFunctionalCluster» State Management

daemon-based

+ CommCtrlRequest()

+ Offer()

+ StopOffer()

«aapPortInterface,aapAPI» CommunicationControl

+ EnableRapidShutdown()

+ ExecuteReset()

+ Offer()

+ RequestReset()

+ StopOffer()

«aapPortInterface,aapAPI» EcuResetRequest

«use»



«aapRequiredPort»

«use»

«aapRequiredPort»



«aapFunctionalCluster»

Diagnostic Management

daemon-based

**Figure 9.92: Interfaces for State Management**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | CommunicationControl | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | DiagnosticComControl | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides functionality for CommunicationControl. | |
| ***Operations:*** | CommCtrlRequest | Called for CommunicationControl (0x28) with any subfunction as subfunction value is part of argument list. |
| Offer | Enable forwarding of request messages from  [Diagnostic Management](#_bookmark214). |
| StopOffer | Disable forwarding of request messages from  [Diagnostic Management](#_bookmark214). |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | EcuResetRequest | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | Yes | |
| ***Meta-model interface type:*** | DiagnosticEcuResetInterface | |
| ***Usage:*** | Public API | |
| ***Description:*** | This interface provides functionality for EcuReset requests. | |
| ***Operations:*** | *EnableRapidShutdown* | Called for subfunction En-/DisableRapidShutdown. |
| ExecuteReset | Execute the requested reset. |

*Δ*

|  |  |  |
| --- | --- | --- |
|  | Offer | Enable forwarding of request messages from  [Diagnostic Management](#_bookmark214). |
| RequestReset | Called for any EcuReset subfunction, except En-/DisableRapidShutdown. |
| StopOffer | Disable forwarding of request messages from  [Diagnostic Management](#_bookmark214). |

* + - * 1. **Interfaces for UDS Transportlayer API**



+ ChannelReestablished()

+ HandleMessage()

+ HandlerStopped()

+ IndicateMessage()

+ NotifyMessageFailure()

+ PeriodicTransmitConfirmation()

+ TransmitConfirmation()

+ GetHandlerID()

+ GetPeriodicHandler()

+ Initialize()

+ NotifyReestablishment()

+ Start()

+ Stop()

+ Transmit()

«aapNativeInterface,aapPlatformExtens...

UdsTransportProtocolMgr

«aapNativeInterface,aapPlatformExtens...

UdsTransportProtocolHandler

UDS Transportlayer Extension

«use» «use»



«aapFunctionalCluster»

Diagnostic Management

daemon-based

**Figure 9.93: Interfaces for the UDS Transportlayer API (1 of 2)**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | UdsTransportProtocolHandler | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Platform extension | |
| ***Description:*** | This interface provides functionality for general Transport Protocol handling. | |
| ***Operations:*** | GetHandlerID | Return the UdsTransportProtocolHandlerID, which was given to the implementation during construction (ctor call). |
| GetPeriodicHandler | Returns the corresponding periodic Transport Protocol handler. |
| Initialize | Initializes the handler. |
| NotifyReestablishment | Notify the [Diagnostic Management](#_bookmark214) core if the given channel has been re-established. |
| Start | Start processing the implemented UDS Transport Protocol. |
| Stop | Indicate that this instance should terminate. |
| Transmit | Transmit a UDS message via the underlying UDS Transport Protocol channel. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | UdsTransportProtocolMgr | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Platform extension | |
| ***Description:*** | This interface provides functionality to manage messages and their handling. | |
| ***Operations:*** | ChannelReestablished | Notification call from the given transport channel, that it has been reestablished since the last (Re)Start from the [UdsTransportProtocolHandler](#_bookmark227) to which this channel belongs. |
| HandleMessage | Hands over a valid received UDS message (currently this is only a request type) from transport layer to session layer. |
| HandlerStopped | Notification from handler, that it has stopped now (e.g. closed down network connections, freed resources, etc...) |
| IndicateMessage | Indicates a message start. |
| NotifyMessageFailure | Indicates, that the message indicated via [IndicateMessage()](#_bookmark231) has failure and will not lead to a final [HandleMessage()](#_bookmark230) call. |
| PeriodicTransmitConfirmation | Confirmation of sent messages and number. |
| TransmitConfirmation | Notification about the outcome of a transmit request called by core [Diagnostic Management](#_bookmark214) at the handler via [Transmit()](#_bookmark228) |



+ GetMaxPayloadLength()

+ GetNumberOfPeriodicMessages()

+ PeriodicTransmit()

+ AddMetaInfo()

+ GetPayload()

+ GetSa()

+ GetTa()

+ GetTaType()

«aapNativeInterface,aapPlatformExtens... UdsTransportProtocolPeriodicHandler

«aapNativeInterface,aapPlatformExtensi...

UdsMessage

UDS Transportlayer Extension

«use» «use»



«aapFunctionalCluster»

Diagnostic Management

daemon-based

**Figure 9.94: Interfaces for the UDS Transportlayer API (2 of 2)**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | UdsMessage | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Platform extension | |
| ***Description:*** | Represents an UDS message exchanged between [Diagnostic Management](#_bookmark214)’s generic core (Uds TransportProtocolMgr) and a specific implementation of UdsTransportProtocolHandler on diagnostic request reception path or diagnostic response transmission path. | |
| ***Operations:*** | AddMetaInfo | Called by the transport plugin to add channel specific meta-info. |
| GetPayload | Get the UDS message data starting with the SID (A\_ Data as per ISO). |
| GetSa | Get the source address of the UDS message. |

*q*

*Δ*

|  |  |  |
| --- | --- | --- |
|  | GetTa | Get the target address of the UDS message. |
| GetTaType | Get the target address type (phys/func) of the UDS message. |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | UdsTransportProtocolPeriodicHandler | |
| ***Technology:*** | Native interface | |
| ***Usage:*** | Platform extension | |
| ***Description:*** | This interface provides functionality for Transport Protocol handling of periodic messages. | |
| ***Operations:*** | GetMaxPayloadLength | Reports the maximum payload length supported for a single periodic transmission on the channel. |
| GetNumberOfPeriodicMessages | Reports the Transport Protocol implementation and connection specific number of periodic messages. |
| PeriodicTransmit | Sends all the messages in the list in the given order. |

* + - * 1. **Interfaces for DoIP API**



+ GetActivationLineState()

+ GetNetworkInterfaceId()

+ Offer()

+ StopOffer()

+ UpdateActivationLineState()

+ GetGidStatus()

+ Offer()

+ StopOffer()

«aapPortInterface,aapPlatformExtension» DoIPActivationLine

«aapPortInterface,aapPlatformExtension» DoIPGroupIdentification

DoIP Extension

«use»

«aapRequiredPort»

«use»

«aapRequiredPort»



«aapFunctionalCluster»

Diagnostic Management

daemon-based

**Figure 9.95: Interfaces for the DoIP API (1 of 2)**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | DoIPGroupIdentification | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | DiagnosticDoIPGroupIdentificationInterface | |
| ***Usage:*** | Platform extension | |
| ***Description:*** | This interface provides functionality to get the GID state of the DoIP protocol. | |
| ***Operations:*** | GetGidStatus | Called to get the current GID state for the DoIP protocol. |
| Offer | Enable forwarding of request messages from  [Diagnostic Management](#_bookmark214). |
| StopOffer | Disables forwarding of request messages from  [Diagnostic Management](#_bookmark214). |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | DoIPActivationLine | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | DiagnosticDoIPActivationLineInterface | |
| ***Usage:*** | Platform extension | |
| ***Description:*** | This interface provides functionality to control the DoIP activation line. | |
| ***Operations:*** | GetActivationLineState | Get the current activation line state. |
| GetNetworkInterfaceId | Get the network interface Id for which this instance is responsible. |
| Offer | Enable provision of the activation line state to  [Diagnostic Management](#_bookmark214). |
| StopOffer | Disable provision of the activation line state to  [Diagnostic Management](#_bookmark214). |
| UpdateActivationLineState | Update current activation line state. |

«use»

«aapRequiredPort»



«aapFunctionalCluster» Diagnostic Management

daemon-based

«use»

«aapProvidedPort»



+ GetDoIPTriggerVehicleAnnouncement()

+ TriggerVehicleAnnouncement()

+ GetDoIPPowerMode()

+ Offer()

+ StopOffer()

«aapPortInterface,aapPlatformExtension» DoIPTriggerVehicleAnnouncement

«aapPortInterface,aapPlatformExtension» DoIPPowerMode

DoIP Extension

**Figure 9.96: Interfaces for the DoIP API (2 of 2)**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | DoIPPowerMode | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | DiagnosticDoIPPowerModeInterface | |
| ***Usage:*** | Platform extension | |
| ***Description:*** | This interface provides functionality to control the power mode via DoIP. | |
| ***Operations:*** | GetDoIPPowerMode | Called to get the current Power Mode for the DoIP protocol. |
| Offer | Enable forwarding of request messages from  [Diagnostic Management](#_bookmark214). |
| StopOffer | Disable forwarding of request messages from  [Diagnostic Management](#_bookmark214). |

|  |  |  |
| --- | --- | --- |
| ***Name:*** | DoIPTriggerVehicleAnnouncement | |
| ***Technology:*** | Port interface | |
| ***Generated:*** | No | |
| ***Meta-model interface type:*** | DiagnosticDoIPTriggerVehicleAnnouncementInterface | |
| ***Usage:*** | Platform extension | |
| ***Description:*** | This interface provides functionality to trigger a vehicle announcement via DoIP. | |
| ***Operations:*** | GetDoIPTriggerVehicleAnnouncement | Get the [DoIPTriggerVehicleAnnouncement](#_bookmark237) interface. |
| TriggerVehicleAnnouncement | Send out vehicle announcements on the given network interface Id. |

* + - 1. **Provided interfaces**

[Diagnostic Management](#_bookmark214) does not provide any interfaces to other Functional Clus- ters.

* + - 1. **Required interfaces**

daemon-based

«aapFunctionalCluster» Diagnostic Management

«use»



«aapRequiredPort»

«use»

«aapRequiredPort»

«use»

«use»



«aapFunctionalCluster» Persistency

+ DiscardPendingChanges()

+ GetAllKeys()

+ GetValue()

+ KeyExists()

+ RecoverKey()

+ RemoveAllKeys()

+ RemoveKey()

+ ResetKey()

+ SetValue()

+ SyncToStorage()

+ DeleteFile()

+ FileExists()

+ GetAllFileNames()

+ GetCurrentFileSize()

+ GetFileInfo()

+ OpenFileReadOnly(): ReadAccessor

+ OpenFileReadWrite(): ReadWriteAccessor

+ OpenFileWriteOnly(): ReadWriteAccessor

+ RecoverFile()

+ ResetFile()

«aapAPI,aapPortInterface» KeyValueStorage

«aapAPI,aapPortInterface» FileStorage



«aapFunctionalClust...

Log and Trace

+ IsEnabled()

+ Log(MsgId, Params)

+ LogDebug(): LogStream

+ LogError(): LogStream

+ LogFatal(): LogStream

+ LogInfo(): LogStream

+ LogVerbose(): LogStream

+ LogWarn(): LogStream

+ WithLevel(): LogStream

«aapAPI,aapNativeInterf...

Logger



«aapFunctionalCluster» Identity and Access Management

+ CheckAccess()

«aapInternal,aapNativeInterface» PolicyDecisionPoint

**Figure 9.97: Interfaces required by Diagnostic Management**

«aapFunctionalCluster» State Management

daemon-based

+ CommCtrlRequest()

+ Offer()

+ StopOffer()

«aapPortInterface,aapAPI» CommunicationControl

+ EnableRapidShutdown()

+ ExecuteReset()

+ Offer()

+ RequestReset()

+ StopOffer()

«aapPortInterface,aapAPI» EcuResetRequest

«use»



«aapRequiredPort»

«use»

«aapRequiredPort»



«aapFunctionalCluster»

Diagnostic Management

daemon-based

**Figure 9.98: Interfaces required by Diagnostic Management from State Management**

«aapInternal» TCP/IP Stack



«aapFunctionalClust... Diagnostic Management daemon-based

«use»



Operating System

**Figure 9.99: Interfaces required by Diagnostic Management from external components**

|  |  |
| --- | --- |
| ***Interface*** | ***Purpose*** |
| TCP/IP Stack | This interface is used for DoIP Transport Protocols. |
| [Diagnostic Management](#_bookmark214)::[Communication](#_bookmark225) [Control](#_bookmark225) | This interface should be used to realize UDS Service 0x28 - CommunicationControl. |
| [Diagnostic Management](#_bookmark214)::[DataElement](#_bookmark221) | This interface is used to handle read DataElement requests. |
| [Diagnostic Management](#_bookmark214)::[DataIdentifier](#_bookmark220) | This interface is used to handle ReadDataByIdentifier and Write DataByIdentifier requests. |
| [Diagnostic Management](#_bookmark214)::[DoIPActivationLine](#_bookmark235) | This interface is used to control a DoIP Transport Layer implementation. |
| [Diagnostic Management](#_bookmark214)::[DoIPGroup](#_bookmark234) [Identification](#_bookmark234) | This interface is used to control a DoIP Transport Layer implementation. |
| [Diagnostic Management](#_bookmark214)::[DoIPPowerMode](#_bookmark236) | This interface is used to control a DoIP Transport Layer implementation. |
| [Diagnostic Management](#_bookmark214)::[DoIPTriggerVehicle](#_bookmark237) [Announcement](#_bookmark237) | This interface is used to control a DoIP Transport Layer implementation. |
| [Diagnostic Management](#_bookmark214)::[DownloadService](#_bookmark224) | This interface is used to handle download requests. |
| [Diagnostic Management](#_bookmark214)::[EcuResetRequest](#_bookmark226) | This interface is used to handle reset requests. |
| [Diagnostic Management](#_bookmark214)::[GenericDataIdentifier](#_bookmark217) | This interface is used to handle ReadDataByIdentifier and Write DataByIdentifier requests. |
| [Diagnostic Management](#_bookmark214)::[GenericRoutine](#_bookmark219) | This interface is used to handle RoutineControl requests. |
| [Diagnostic Management](#_bookmark214)::[GenericUDSService](#_bookmark218) | This interface is used to handle UDS requests. |
| [Diagnostic Management](#_bookmark214)::[Routine](#_bookmark222) | This interface is used to handle RoutineControl requests. |
| [Diagnostic Management](#_bookmark214)::[SecurityAccess](#_bookmark215) | This interface is used to handle SecurityAccess requests. |

*q*

*Δ*

|  |  |
| --- | --- |
| ***Interface*** | ***Purpose*** |
| [Diagnostic Management](#_bookmark214)::[ServiceValidation](#_bookmark216) | This interface is used to handle ServiceValidation requests. |
| [Diagnostic Management](#_bookmark214)::[UdsMessage](#_bookmark232) | This interface is used to access an UDS Transport Layer implementation. |
| [Diagnostic Management](#_bookmark214)::[UdsTransportProtocol](#_bookmark227) [Handler](#_bookmark227) | This interface is used to access an UDS Transport Layer implementation. |
| [Diagnostic Management](#_bookmark214)::[UdsTransportProtocol](#_bookmark229) [Mgr](#_bookmark229) | This interface is used to access an UDS Transport Layer implementation. |
| [Diagnostic Management](#_bookmark214)::[UdsTransportProtocol](#_bookmark233) [PeriodicHandler](#_bookmark233) | This interface is used to access an UDS Transport Layer implementation. |
| [Diagnostic Management](#_bookmark214)::[UploadService](#_bookmark223) | This interface is used to handle upload requests. |
| [Execution Management](#_bookmark61)::[ExecutionClient](#_bookmark64) | This interface is used to report the status of the [Diagnostic](#_bookmark214) [Management](#_bookmark214) daemon process(es). |
| [Identity and Access Management](#_bookmark175)::[Policy](#_bookmark176) [DecisionPoint](#_bookmark176) | [Diagnostic Management](#_bookmark214) shall use this interface to check access from  Diagnostic Clients. |
| [Log and Trace](#_bookmark90)::[Logger](#_bookmark91) | [Diagnostic Management](#_bookmark214) shall use this interface to log standardized messages. |
| [Persistency](#_bookmark124)::[FileStorageOperations](#_bookmark127) | This interface is used to read and write data to files. |
| [Persistency](#_bookmark124)::[KeyValueStorageOperations](#_bookmark138) | This interface should be used to persist key-value data. |
| [Persistency](#_bookmark124)::[PersistencyHandlingOperations](#_bookmark147) | This interface is used for general persistency handling. |
| [Persistency](#_bookmark124)::[FileStorage](#_bookmark130) | This interface is used to read and write data to files. |
| [Persistency](#_bookmark124)::[KeyValueStorage](#_bookmark141) | This interface should be used to persist key-value data. |
| [Persistency](#_bookmark124)::[ReadAccessor](#_bookmark134) | This interface is used to read and write data to files. |
| [Persistency](#_bookmark124)::[ReadWriteAccessor](#_bookmark135) | This interface is used to read and write data to files. |
| [Platform Health Management](#_bookmark186)::[SupervisedEntity](#_bookmark189) | This interface is used to supervise the [Diagnostic Management](#_bookmark214)  daemon process(es). |
| [Registry](#_bookmark210)::[ManifestAccessor](#_bookmark211) | [Diagnostic Management](#_bookmark214) shall use this interface to read its configuration information from the Manifests. |
| [State Management](#_bookmark76)::[DiagnosticReset](#_bookmark80) | This interface is used to handle diagnostic reset requests. |

**Table 9.28: Interfaces required by Diagnostic Management**

# Use-Case View

This chapter provides an overview of the use cases of the AUTOSAR Adaptive Plat- form. The use cases are defined on the level of the Functional Clusters in the AUTOSAR Adaptive Platform. The chapter structure corresponds to the Building Block View in Section [9](#_bookmark55).

The use cases in this chapter are specified in a solution-neutral way. As a conse- quence, include and extend relationships are used sparingly to avoid "programming" with use cases. Please refer to the individual scenarios in chapter [11](#_bookmark257) that provide a detailed insight how the use cases could be implemented and how they refer to each other.

The use cases are described using tables that are based on the template shown in Table [10.1](#_bookmark239). Some rows are optional and will be left out if empty.

|  |  |  |
| --- | --- | --- |
| ***Name:*** | The name of this use case | |
| ***Description:*** | A brief description of this use case. | |
| ***Supporting:*** | If yes, this use case is defined just for factoring out common behavior by means of include or extend relationships only. The use case is not triggered directly by an actor and usually does not have own scenarios. This row is optional. | |
| ***Triggering Actors:*** | The name of an actor that triggers this use case. This row is optional. | A brief description how the actor triggers the use case. |
| ***Participating Actors:*** | The name of an actor that participates in this use case. This row is optional. | A brief description how the actor participates the use case. |
| ***Base use case:*** | The name of the use case that this use cases specializes. This row is optional. | |
| ***Included use cases:*** | The name of an use case that is included by this use case. This row is optional. | A brief description of the reason for including the use case. |
| ***Extending use cases:*** | The name of an use case that extends this use case. This row is optional. | A brief description of the condition for extending this use case. |
| ***Preconditions:*** | A list of preconditions that need to be fulfilled before the use case can be executed. | |
| ***Invariants:*** | A list of invariants that need to be fulfilled before the use case can be executed, while the use case is executed, and after the use case has been executed. | |
| ***Postconditions:*** | A list of postconditions that need to be fulfilled after the use case has been executed. | |
| ***Scenarios:*** | The name of the scenario. This row is optional. | A brief description of the scenario. |

**Table 10.1: Template for Use Cases**

## Runtime

### Execution Management

Adaptive Platform



Execution Management

«aapUseCase» Start Adaptive Platform

«aapFunctionalCluster»

Platform Health Management

daemon-based

»

«aapTriggers»

Adaptive Application

*(from*

«include»

«aapUseCase»

«aapParticipates

*Actors)*

«aapTriggers»

Change Function Group State

«aa

«aapFunctionalCluster»

State Management

daemon-based

pTriggers»

«aapParticipates»

«include»

«aapPart Operating System

«aap

Triggers»

Shutdown Adaptive Platform

«aapUseCase»

icipates»

*(from Actors)*

**Figure 10.1: Use cases for Execution Management**

* + - 1. **Start Adaptive Platform**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | Start Adaptive Platform | |
| ***Description:*** | This use case describes the start of the AUTOSAR Adaptive Platform. | |
| ***Triggering Actors:*** | Operating System | The [Operating System](#_bookmark28) triggers this use case by starting [Execution](#_bookmark61) [Management](#_bookmark61) as the first process of the AUTOSAR Adaptive Platform. The [Operating System](#_bookmark28) also participates in this use case by starting processes. |
| ***Included use cases:*** | [Change Function Group](#_bookmark244) [State](#_bookmark244) | This use case is included to perform the transition to the Machine Function Group State named Startup. |
| ***Preconditions:*** | * The base software, for example a operating system or a hypervisor, is initialized and running. | |
| ***Invariants:*** | * The configuration of the base software remains unchanged, in particular the processes of base software. | |
| ***Postconditions:*** | * The standardized Machine Function Group State named   Startup has been entered. | |

*q*

*Δ*

|  |  |  |
| --- | --- | --- |
| ***Scenarios:*** | [Start Platform with](#_bookmark260) [Supervision of](#_bookmark260) [Applications](#_bookmark260) | This scenario shows the startup of the Adaptive Platform with an supervision of Adaptive Applications. It therefore includes the startup of [Platform](#_bookmark186) [Health Management](#_bookmark186) that performs supervision and the startup of an Adaptive Application that is supervised as part of the Startup Machine Function Group State. Additional Functional Clusters and Application processes may need to be started in addition depending on the Adaptive Platform implementation and project-specific needs. |

**Table 10.2: Use-Case Start Adaptive Platform**

* + - 1. **Shutdown Adaptive Platform**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | Shutdown Adaptive Platform | |
| ***Description:*** | This use case describes the shutdown of the AUTOSAR Adaptive Platform. | |
| ***Triggering Actors:*** | State Management | [State Management](#_bookmark76) triggers this use case by requesting the standardized Machine Function Group State named Shutdown. |
| ***Participating Actors:*** | Operating System | The [Operating System](#_bookmark28) participates in this use case by starting and stopping processes. |
| ***Included use cases:*** | [Change Function Group](#_bookmark244) [State](#_bookmark244) | This use case is included to perform the transition to the Machine Function Group State named Shutdown. |
| ***Preconditions:*** | None | |
| ***Invariants:*** | None | |
| ***Postconditions:*** | * The AUTOSAR Adaptive Platform has been shutdown. | |
| ***Scenarios:*** | [Shutdown Platform with](#_bookmark262) [Supervision of](#_bookmark262) [Applications](#_bookmark262) | This scenario shows the regular shutdown of the Adaptive Platform. It includes the shutdown of [Platform Health](#_bookmark186) [Management](#_bookmark186) that performs supervision of applications. |

**Table 10.3: Use-Case Shutdown Adaptive Platform**

* + - 1. **Change Function Group State**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | Change Function Group State | |
| ***Description:*** | This use case describes the change of a Function Group State. The use case is applicable to both, Function Group States defined by an application as well as predefined Function Group States of the AUTOSAR Adaptive Platform (i.e., Machine Function Group State).  See Sections [13.2](#_bookmark300) and [13.3](#_bookmark301) for details. | |
| ***Triggering Actors:*** | State Management | [State Management](#_bookmark76) triggers this use case by requesting a Function Group State from [Execution Management](#_bookmark61). |
| ***Participating Actors:*** | Operating System | The Operating System participates in this use case by starting / stopping processes as requested by [Execution](#_bookmark61) [Management](#_bookmark61). |
| Adaptive Application | The [Adaptive Application](#_bookmark24) participates in this use case by reporting its execution state to [Execution](#_bookmark61) [Management](#_bookmark61). |
| Platform Health Management | Platform Health Management participates in this use case by using information from [Execution](#_bookmark61) [Management](#_bookmark61) about processes to be newly started or terminated for coordination with supervision checkpoints. |
| ***Preconditions:*** | None | |
| ***Invariants:*** | * The configuration of the base software remains unchanged, in particular the processes of base software. | |
| ***Postconditions:*** | * The requested Function Group State has been entered successfully. | |
| ***Scenarios:*** | [Change Function Group State](#_bookmark264) | This scenario shows the regular change of a Function Group State. |

**Table 10.4: Use-Case Change Function Group State**

### State Management



Adaptive Application

**Figure 10.2: Use cases for State Management**



Adaptive Platform

State Management

«aapUseCase»

«aapTriggers»

Change System State

«aapFunctionalClust...

«aapParticipates» Execution Management

daemon-based

«include»

«aapUseCase»

Recover from Supervision Failure

«aapTriggers»

«aapFunctionalClust...

Platform Health Management

daemon-based

* + - 1. **Change System State**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | Change System State | |
| ***Description:*** | This use case describes the change the state of the system (i.e., the configuration and extent of the running Processes). | |
| ***Triggering Actors:*** | Adaptive Application | The [Adaptive Application](#_bookmark24) triggers this use case by changing the value of the field Trigger in a TriggerIn\_{StateGroup} or TriggerInOut\_{StateGroup} service interface. |
| ***Participating Actors:*** | Execution Management | [Execution Management](#_bookmark61) participates in this use case by requesting a start or termination of processes from the Operating System. |
| ***Preconditions:*** | * The system is not in Machine State Shutdown. | |
| ***Invariants:*** | None | |
| ***Postconditions:*** | * The system state (i.e. the configuration and extent of running   Processes) has changed. | |
| ***Scenarios:*** | [Accept Trigger Input](#_bookmark267) | This scenario shows a state change reaction by [State Management](#_bookmark76) after a change to a TriggerIn\_{StateGroup} or TriggerInOut\_{StateGroup} service interface. |
| [Reject Trigger Input](#_bookmark269) | This scenario demonstrates how a change to a TriggerIn\_{StateGroup} or TriggerInOut\_{StateGroup} service interface does not lead to a state change reaction by [State Management](#_bookmark76). |

**Table 10.5: Use-Case Change System State**

* + - 1. **Perform Recovery**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | Recover from Supervision Failure | |
| ***Description:*** | This use case describes the recovery from a supervision failure reported by  [Platform Health Management](#_bookmark186). | |
| ***Triggering Actors:*** | Platform Health Management | [Platform Health Management](#_bookmark186)  triggers this use case. |
| ***Included use cases:*** | [Change System State](#_bookmark246) | This use case is included to perform recovery by requesting a transition to a new system state. |

*q*

*Δ*

|  |  |  |
| --- | --- | --- |
| ***Preconditions:*** | * [State Management](#_bookmark76) needs to be registered with the   [RecoveryAction](#_bookmark192) interface of [Platform Health Management](#_bookmark186). | |
| ***Invariants:*** | None | |
| ***Postconditions:*** | No postconditions defined at the moment. | |
| ***Scenarios:*** | [Successful Recovery](#_bookmark271) | Default scenario that shows a successful recovery from a supervision failure by switching to a new system state. |

**Table 10.6: Use-Case Recover from Supervision Failure**

## Storage

### Persistency

The use cases for [Persistency](#_bookmark124) are organized into three categories. Section [10.2.1.1](#_bookmark249) lists the use cases for reading persistent data. Section [10.2.1.2](#_bookmark252) lists the use cases for storing persistent data. Section [10.2.1.3](#_bookmark254) lists the use cases related to monitoring of persistent storage.

* + - 1. **Reading Persistent Data**

Adaptive Platform



Persistency

Adaptive Application

*(from Actors)*

«aapTriggers»

«aapUseCase»

*Read Persistent Data*

«aapUseCase» Read Persistent Data with Unique ID



Operating System

*(from Actors)*

«aapParticipates»

«extend»

«aapUseCase» Detect and Correct Data

Errors

«extend»

«aapUseCase» Read Persistent Data from File

«aapUseCase» Decrypt Persistent Data

«aapParticipates»

«aapFunctionalCluster»

Cryptography

daemon-based

**Figure 10.3: Use cases for Reading Persistent Data**

* + - * 1. **Read Persistent Data from File**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | Read Persistent Data from File | |
| ***Description:*** | This use case describes reading persistent data from a file. | |
| ***Triggering Actors:*** | Adaptive Application | The [Adaptive Application](#_bookmark24) triggers this use case. |
| ***Participating Actors:*** | Operating System | The [Operating System](#_bookmark28) participates in this use case by providing access to the underlying storage. |
| ***Base use case:*** | Read Persistent Data | |
| ***Extending use cases:*** | [Detect and Correct Data](#_bookmark251) [Errors](#_bookmark251) | [Detect and Correct Data Errors](#_bookmark251) extends this use case if redundancy is configured in the Manifest (see PersistencyRedundancyHandling). |
| [Decrypt Persistent Data](#_bookmark250) | [Decrypt Persistent Data](#_bookmark250) extends this use case if encryption is configured in the Manifest (see PersistencyDe- ploymentToCryptoKeySlotMapping or PersistencyDeploymentElement- ToCryptoKeySlotMapping). |
| ***Preconditions:*** | * The requested file exists. | |
| ***Invariants:*** | None | |
| ***Postconditions:*** | None | |
| ***Scenarios:*** | [Read Data from File](#_bookmark275) [Successfully](#_bookmark275) | This scenario shows how data is read from a file successfully. |

**Table 10.7: Use-Case Read Persistent Data from File**

* + - * 1. **Read Persistent Data with Unique ID**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | Read Persistent Data With Unique ID | |
| ***Description:*** | This use case describes reading persistent data associated with a unique ID (key). | |
| ***Triggering Actors:*** | Adaptive Application | The [Adaptive Application](#_bookmark24) triggers this use case. |
| ***Participating Actors:*** | Operating System | The Operating System participates in this use case by providing access to the underlying storage. |
| ***Base use case:*** | Read Persistent Data | |
| ***Extending use cases:*** | [Detect and Correct Data](#_bookmark251) [Errors](#_bookmark251) | [Detect and Correct Data Errors](#_bookmark251) extends this use case if redundancy is configured in the Manifest (see PersistencyRedundancyHandling). |

*q*

*Δ*

|  |  |  |
| --- | --- | --- |
|  | [Decrypt Persistent Data](#_bookmark250) | [Decrypt Persistent Data](#_bookmark250) extends this use case if encryption is configured in the Manifest (see PersistencyDe- ploymentToCryptoKeySlotMapping or PersistencyDeploymentElement- ToCryptoKeySlotMapping). |
| ***Preconditions:*** | * The requested unique ID (key) exists. | |
| ***Invariants:*** | None | |
| ***Postconditions:*** | None | |
| ***Scenarios:*** | [Read Data with Unique ID](#_bookmark277) [Successfully](#_bookmark277) | This scenario shows how data is read for a unique ID successfully. |

**Table 10.8: Use-Case Read Persistent Data With Unique ID**

* + - * 1. **Decrypt Persistent Data**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | Decrypt Persistent Data | |
| ***Description:*** | This use case describes decryption of persistent data. | |
| ***Supporting:*** | Yes | |
| ***Participating Actors:*** | Cryptography | [Cryptography](#_bookmark150) participates in this use case by decrypting data. |
| ***Preconditions:*** | None | |
| ***Invariants:*** | None | |
| ***Postconditions:*** | None | |

**Table 10.9: Use-Case Decrypt Persistent Data**

* + - * 1. **Detect and Correct Data Errors**

|  |  |
| --- | --- |
| ***Name:*** | Detect and Correct Data Errors |
| ***Description:*** | This use case describes detection of errors in persistent data and their correction. |
| ***Supporting:*** | Yes |
| ***Preconditions:*** | None |
| ***Invariants:*** | None |
| ***Postconditions:*** | None |

* + - 1. **Storing Persistent Data**

Adaptive Platform



Persistency

Adaptive Application

*(from Actors)*

«aapTriggers»

«aapUseCase»

*Store Persistent Data*

«aapUseCase» Store Persistent Data with Unique ID



Operating System

*(from Actors)*

«aapParticipates»

«extend»

«aapUseCase» Encrypt Persistent Data

«aapUseCase» Store Persistent Data in

File

«aapParticipates»

«aapFunctionalCluster»

Cryptography

daemon-based

**Figure 10.4: Use cases for Storing Persistent Data**

* + - * 1. **Store Persistent Data in File**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | Store Persistent Data in File | |
| ***Description:*** | This use case describes storing persistent data to a file. | |
| ***Triggering Actors:*** | Adaptive Application | The [Adaptive Application](#_bookmark24) triggers this use case. |
| ***Participating Actors:*** | Operating System | The [Operating System](#_bookmark28) participates in this use case by providing access to the underlying storage. |
| ***Base use case:*** | Store Persistent Data | |
| ***Extending use cases:*** | [Encrypt Persistent Data](#_bookmark253) | [Encrypt Persistent Data](#_bookmark253) extends this use case if encryption is configured in the Manifest (see PersistencyDe- ploymentToCryptoKeySlotMapping or PersistencyDeploymentElement- ToCryptoKeySlotMapping). |
| ***Preconditions:*** | * The requested file exists. | |
| ***Invariants:*** | None | |
| ***Postconditions:*** | None | |
| ***Scenarios:*** | [Store Data to File](#_bookmark279) [Successfully](#_bookmark279) | This scenario shows how data is successfully stored to a file. |

* + - * 1. **Store Persistent Data with Unique ID**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | Store Persistent Data with Unique ID | |
| ***Description:*** | This use case describes storing persistent data associated with a unique ID (key). | |
| ***Triggering Actors:*** | Adaptive Application | The [Adaptive Application](#_bookmark24) triggers this use case. |
| ***PArticipating Actors:*** | Operating System | The [Operating System](#_bookmark28) participates in this use case by providing access to the underlying storage. |
| ***Base use case:*** | Store Persistent Data | |
| ***Extending use cases:*** | [Encrypt Persistent Data](#_bookmark253) | [Encrypt Persistent Data](#_bookmark253) extends this use case if encryption is configured in the Manifest (see PersistencyDe- ploymentToCryptoKeySlotMapping or PersistencyDeploymentElement- ToCryptoKeySlotMapping). |
| ***Preconditions:*** | * The requested unique ID (key) exists. | |
| ***Invariants:*** | None | |
| ***Postconditions:*** | None | |
| ***Scenarios:*** | [Store Data with Unique](#_bookmark281) [ID Sucessfully](#_bookmark281) | This scenario shows how data is successfully stored with a unique ID (key). |

**Table 10.12: Use-Case Store Persistent Data with Unique ID**

* + - * 1. **Encrypt Persistent Data**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | Encrypt Persistent Data | |
| ***Description:*** | This use case describes the encryption of persistent data. | |
| ***Supporting:*** | Yes | |
| ***Participating Actors:*** | Cryptography | The [Cryptography](#_bookmark150) participates in this scenario by encrypting persistent data. |
| ***Preconditions:*** | None | |
| ***Invariants:*** | None | |
| ***Postconditions:*** | None | |

* + - 1. **Monitoring**

Adaptive Platform

Persistency

Adaptive Application

*(from Actors)*

«aapTriggers»

«aapParticipates»

«aapUseCase» Get Storage Size

Operating System

*(from Actors)*

**Figure 10.5: Use cases for Monitoring**

* + - * 1. **Get Storage Size**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | Get Storage Size | |
| ***Description:*** | This use case describes determining the size of the storage space occupied by persistent data. | |
| ***Triggering Actors:*** | Adaptive Application | The [Adaptive Application](#_bookmark24) triggers this use case. |
| ***Participating Actors:*** | Operating System | The [Operating System](#_bookmark28) participates in this use case by providing information from underlying storage. |
| ***Preconditions:*** | None | |
| ***Invariants:*** | None | |
| ***Postconditions:*** | None | |
| ***Scenarios:*** | [Get FileStorage Size](#_bookmark283) | This scenario shows how to obtain information about occupied storage space of a [FileStorage](#_bookmark130). |
| [Get KeyValueStorage Size](#_bookmark285) | This scenario shows how to obtain information about occupied storage space of a [KeyValueStorage](#_bookmark141). |

**Table 10.14: Use-Case Get Storage Size**

## Security

### Firewall





Adaptive Application

*(from Actors)*

«aapTriggers»

Firewall

«aapUseCase» Switch Firewall State

Adaptive Platform

«aapParticipates»

Operating System

*(from Actors)*

«aapTriggers»

«aapUseCase» Report Security Event

«aapParticipates»

«aapFunctionalCluster» Adaptive Intrusion Detection System

Manager daemon-based

**Figure 10.6: Use cases for Firewall**

* + - 1. **Switch Firewall State**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | Switch Firewall State | |
| ***Description:*** | This use case describes switching the firewall state. | |
| ***Triggering Actors:*** | Adaptive Application | The [Adaptive Application](#_bookmark24) triggers this use case. |
| ***Participating Actors:*** | Operating System | The [Operating System](#_bookmark28) participates in this use case by applying the firewall rules associated with the target firewall state to its TCP/IP stack or, for example, a hardware firewall. |
| ***Preconditions:*** | None | |
| ***Invariants:*** | None | |
| ***Postconditions:*** | * The active firewall state has changed to the requested target state. * The configured firewall rules have been applied to the underlying firewall implementation. | |
| ***Scenarios:*** | [Switch Firewall State](#_bookmark289) [Successfully](#_bookmark289) | This scenario shows the successful switch of the firewall state. |

**Table 10.15: Use-Case Switch Firewall State**

* + - 1. **Report Security Event**

|  |  |  |
| --- | --- | --- |
| ***Name:*** | Report Security Event | |
| ***Description:*** | This use case describes reporting a Security Event to [Adaptive](#_bookmark178) [Intrusion Detection System Manager](#_bookmark178) if a message is received that matches a configured rule for blocked messages. | |
| ***Triggering Actors:*** | Operating System | The [Operating System](#_bookmark28) triggers this use case by informing the [Firewall](#_bookmark182) that a message was received that matches a configured rule for blocked messages. |
| ***Participating Actors:*** | Adaptive Intrusion Detection System Manager | [Adaptive Intrusion Detection](#_bookmark178) [System Manager](#_bookmark178) participates in this use case by handling the generated Security Event. |
| ***Preconditions:*** | * A rule for blocked messages and a corresponding Security Event   have been configured in the Manifest. | |
| ***Invariants:*** | None | |
| ***Postconditions:*** | * A Security Event has been generated and forwarded to the   [Adaptive Intrusion Detection System Manager](#_bookmark178). | |
| ***Scenarios:*** | [Report Security Event](#_bookmark291) [Successfully](#_bookmark291) | This scenario shows the successful reporting of an security event. |

**Table 10.16: Use-Case Report Security Event**

# Runtime View

This chapter shows the original design approach of the AUTOSAR Adaptive Platform for implementing selected use cases. The presented use cases currently cover just a small part of the functionality of the AUTOSAR Adaptive Platform. More use cases will be added in future versions of this document. Please note that individual imple- mentations of the AUTOSAR Adaptive Platform may always choose a different design internally. Thus, interaction partners, the type of messages, and their order may differ.

* 1. **Runtime**
     1. **Execution Management**
        1. **Start Adaptive Platform**
           1. **Scenario: Start Platform with Supervision of Applications**



:Operating System

:WatchdogInterface

«aapFunctionalCluster»

:Core

enable()

«aapFunctionalClust...

:Execution Management

main()

SetState(Startup)

exec()

«aapFunctionalClust...

:State Management

main()

ReportExecutionState(kRunning): Result

opt

exec()

«aapFunctionalCluster»

:Platform Health Management

main()

AliveNotification()

ReportExecutionState(kRunning): Result

Offer()

loop Supervision Loop

performSupervision() AliveNotification()

opt

exec()

:Adaptive Application

main()

Initialize()

ReportExecutionState(kRunning): Result

ReportCheckpoint()

GetInitialMachineStateTransitionResult()

ref

Change Function Group State (Running)

**Figure 11.1:** **Scenario: Start Platform with Supervision of Applications**

Figure [11.1](#_bookmark261) shows a scenario of for [Start Adaptive Platform](#_bookmark242) with an supervi- sion of Adaptive Applications. It therefore includes the startup of [Platform Health](#_bookmark186) [Management](#_bookmark186) that performs supervision and the startup of an Adaptive Application that is supervised as part of the Startup Machine Function Group State.

During the preceding startup of the Machine the [Operating System](#_bookmark28) performs ini- tialization steps in an implementation-specific way. These steps include starting any middleware related to the [Operating System](#_bookmark28), including device-drivers and services handling low-level middleware. The AUTOSAR Adaptive Platform demands that the

[Watchdog](#_bookmark30) is enabled prior to the startup of the AUTOSAR Adaptive Platform, for ex- ample, the [Watchdog](#_bookmark30) could already be enabled by the Bootloader or the [Operat-](#_bookmark28) [ing System](#_bookmark28).

[Execution Management](#_bookmark61) is started by the [Operating System](#_bookmark28) as the first process of the AUTOSAR Adaptive Platform. [Execution Management](#_bookmark61) then controls the startup of the AUTOSAR Adaptive Platform by activating the standardized Function Group State called Startup of the Machine Function Group State. This trig- gers the start of additional processes that are configured to run in the Startup state. It is mandatory that [State Management](#_bookmark76) is part of the Startup state. Other processes of the AUTOSAR Adaptive Platform, for example [Platform Health Management](#_bookmark186) and application processes may also be part of the Startup state (see Figure [11.1](#_bookmark261)).

[Platform Health Management](#_bookmark186) is responsible to service the [Watchdog](#_bookmark30). Thus, the time between enabling the [Watchdog](#_bookmark30) during the start of the Machine and the start of [Platform Health Management](#_bookmark186) needs to be less than the [Watchdog](#_bookmark30) timeout. The integrator needs to fulfill this constraint.

After the Startup state has been reached, [State Management](#_bookmark76) takes over control to determine the desired Function Group States.

* + - 1. **Shutdown Adaptive Platform**
         1. **Scenario: Shutdown Platform with Supervision of Applications**



«aapFunctionalClust... «aapFunctionalClust... «aapFunctionalCluster» :Operating System

:State Management :Execution Management :Platform Health Management

StopOffer()

ref

Change Function Group State (Shutdown)

SIGTERM()

exec()

shutdownApp: Adaptive Application

main()

shutdown()

SIGTERM()

SIGTERM()

SIGTERM()

**Figure 11.2:** **Scenario: Shutdown Platform with Supervision of Applications**

Figure [11.2](#_bookmark263) shows a scenario for [Shutdown Adaptive Platform](#_bookmark243) with a super- vision of Adaptive Applications. It therefore includes the shutdown of [Platform](#_bookmark186) [Health Management](#_bookmark186) that performs supervision.

The shutdown is triggered by [State Management](#_bookmark76) by requesting the standardized Machine Function Group State called Shutdown. In general, it is assumed that the only processes configured to run in the Shutdown state are [State Management](#_bookmark76) and an application that issues a shutdown request towards the [Operating System](#_bookmark28) (shutdownApp in Figure [11.2](#_bookmark263)). [Execution Management](#_bookmark61) will therefore perform an orderly shutdown of the other running application and platform processes (including [Platform Health Management](#_bookmark186)) before starting the application process that issues a shutdown request towards the [Operating System](#_bookmark28). The [Operating System](#_bookmark28) terminates the remaining processes (i.e. [State Management](#_bookmark76), [Execution Man-](#_bookmark61) [agement](#_bookmark61)) of the AUTOSAR Adaptive Platform and shuts down the Machine in an implementation-specific way.

* + - 1. **Change Function Group State**
         1. **Scenario: Change Function Group State**



«aapFunctionalClust... «aapFunctionalClust... :Operating System old: Adaptive Application 0..\* «aapFuncti... «aapFunctionalCluster»

:State Management :Execution Management :Core :Platform Health Management

SetState(FunctionGroupState)

startStateTransition(old, new)

determineObsoleteProcesses()

loop Terminate obsolete Processes

processTerminating(process)

stopSupervision()

SIGTERM()

Deinitialize()

determineProcessesToStart()

loop Start Processes

exec()

new: Adaptive Application 0..\*

main()

Initialize()

ReportExecutionState(kRunning)

processStarted(process, timestamp)

startAliveSupervision()

ReportCheckpoint()

**Figure 11.3:** **Scenario: Regular change of a Function Group State**

Figure [11.3](#_bookmark265) shows a scenario for changing a Function Group State. The sce- nario is triggered by [State Management](#_bookmark76). [Execution Management](#_bookmark61) will terminate all processes that are not part of the requested target Function Group State or that have a different StateDependentStartupConfig.

Just before terminating a process (with a SIGTERM signal), [Execution Manage-](#_bookmark61) [ment](#_bookmark61) notifies [Platform Health Management](#_bookmark186) that will stop all supervisions of the process. Consequently, [State Management](#_bookmark76) will not receive any information about failed supervisions during the shutdown of the process. The shutdown is monitored by [Execution Management](#_bookmark61) by means of StartupConfig.timeout configured in the Manifest.

Afterwards, [Execution Management](#_bookmark61) starts the processes of the target Function Group State in the order imposed by their StateDependentStartupConfig. When a processes reports its execution state as kRunning, [Execution Manage-](#_bookmark61) [ment](#_bookmark61) notifies [Platform Health Management](#_bookmark186) to start Alive Supervision for

that process. Other kinds of supervisions (Deadline Supervision, and Logical Supervision) are started when the first checkpoint is reported for them.

* + 1. **State Management**
       1. **Change System State**
          1. **Scenario: Accept Trigger Input**

«aapFunctionalClust...



«aapFunctionalClust...

:Execution Management

:State Management

app: Adaptive Application

TriggerIn\_{StateGroup}::Set(value)

determineDesiredFGSate(): desiredState

SetState(desiredState)

handleNewFGState()

TriggerOut\_{StateGroup}::notify(value)

**Figure 11.4:** **Scenario: Accept Trigger Input**

Figure [11.4](#_bookmark268) shows the default scenario for changing the system state. An [Adaptive](#_bookmark24) [Application](#_bookmark24) changes a field in the TriggerIn\_{StateGroup} service interface. Alternatively, the TriggerInOut\_{StateGroup} may be used (not shown). Based on the new input data, [State Management](#_bookmark76) determines a new desired system state, that is a set of desired Function Group States, and requests these Function Group States from [Execution Management](#_bookmark61) by calling [SetState()](#_bookmark72).

After the new system state has been reached, [State Management](#_bookmark76) updates the fields in the TriggerOut\_{StateGroup} (and TriggerInOut\_{StateGroup}, not shown) interfaces accordingly.

* + - * 1. **Scenario: Reject Trigger Input**



app: Adaptive Application



determineDesiredFGState(): desiredState

{desiredState = currentState}

TriggerIn\_{StateGroup}::Set(value)

«aapFunctionalClust...

:State Management

**Figure 11.5:** **Scenario: Reject Trigger Input**

Figure [11.5](#_bookmark270) shows an alternate scenario. An [Adaptive Application](#_bookmark24) changes a field in the TriggerIn\_{StateGroup} (alternatively TriggerI- nOut\_{StateGroup}) service interface. Despite the new input data, [State Man-](#_bookmark76) [agement](#_bookmark76) determines that the current system state is still the desired system state. Therefore, no further action is taken by [State Management](#_bookmark76).

* + - 1. **Recover from Supervision Failure**
         1. **Scenario: Successful Recovery**



supervisionFailed()

handleFailedSupervision()

RecoveryHandler()

determineDesiredFGState(): desiredState

ref

Change System State

Offer()

«aapFunctionalCluster»

:Execution Management

«aapFunctionalCluster»

:State Management

«aapFunctionalCluster»

:Platform Health Management

**Figure 11.6:** **Scenario: Successful Recovery**

Figure [11.6](#_bookmark272) shows the scenario for performing successful recovery after a supervision failure has been detected by [Platform Health Management](#_bookmark186). [Platform Health](#_bookmark186) [Management](#_bookmark186) notifies [State Management](#_bookmark76) by invoking the call-back method [Recov-](#_bookmark194) [eryHandler()](#_bookmark194). [State Management](#_bookmark76) then determines a new desired state (which may be the same as the current state) and requests corresponding Function Group State transitions from [Execution Management](#_bookmark61).

* 1. **Storage**
     1. **Persistency**
        1. **Read Persistent Data from File**
           1. **Scenario: Read Data from File Successfully**



«aapFunctionalCluster»

:Persistency

«aapFunctionalCluster»

:Cryptography

:Adaptive Application

:Operating System

OpenFileStorage(): FileStorage

readAccessor= OpenFileReadOnly(): ReadAccessor

open()

alt

[Redundancy enabled in Manifest] ref

Detect and Correct Data Errors

alt

[KeySlotMapping exists in Manifest and keySlotUsage = encryption]

ref

Initialize Cipher

loop

[not readAccessor.IsEof()]

line= ReadLine()

loop

read()

alt

[KeySlotMapping exists in Manifest and keySlotUsage = encryption]

ProcessBlock()

**Figure 11.7:** **Scenario: Read Data from File Successfully**

Figure [11.7](#_bookmark276) shows the scenario for successfully reading data from a file successfully. The [Adaptive Application](#_bookmark24) needs to open a [FileStorage](#_bookmark130) by calling [OpenFileStor-](#_bookmark129) [age()](#_bookmark129). Afterwards, the [Adaptive Application](#_bookmark24) needs to open an individual file by calling [OpenFileReadOnly()](#_bookmark131) or [OpenFileReadWrite()](#_bookmark132) depending if the [Adaptive Ap-](#_bookmark24) [plication](#_bookmark24) needs to write data to the file as well. Then, the [Adaptive Applica-](#_bookmark24) [tion](#_bookmark24) can read data via the methods provided by [ReadAccessor](#_bookmark134) either as binary data or text data.

If enabled in the Manifest, [Persistency](#_bookmark124) will detect and try to correct data errors when reading from a file. If the file or [FileStorage](#_bookmark130) is configured to use encryption, the contents of the file will be transparently decrypted during read.

* + - 1. **Read Persistent Data with Unique ID**
         1. **Scenario: Read Data with Unique ID Successfully**



:Adaptive Application :Operating System

**Figure 11.8:** **Scenario: Read Data with Unique ID Successfully**



OpenKeyValueStorage(): KeyValueStorage

open()

read()

alt

[KeySlotMa

keySlotUsage = encryption]

sBlock()

Proces

ze Cipher

Initiali

ref

ing exists in Manifest and

pp

rrors

Detect and Correct Data E

ref

enabled in the Manifest]

y

alt

[Redundanc

GetValue(key)

«aapFunctionalClust...

:Cryptography

«aapFunctionalClust...

:Persistency

Figure [11.8](#_bookmark278) shows the scenario for successfully reading data by providing a unique identifier (key). The [Adaptive Application](#_bookmark24) needs to open a [KeyValueStorage](#_bookmark141) by calling [OpenKeyValueStorage()](#_bookmark140). Then, the [Adaptive Application](#_bookmark24) can read data associated to a key by calling [GetValue()](#_bookmark142).

If enabled in the Manifest, [Persistency](#_bookmark124) will detect and try to correct data errors when reading data. If the individual key or [KeyValueStorage](#_bookmark141) is configured to use en- cryption, data will be transparently decrypted during read.

* + - 1. **Store Persistent Data to File**
         1. **Scenario: Store Data to File Successfully**



:Adaptive Application

«aapFunctionalClust...

:Persistency





:Operating System

«aapFunctionalClust...

:Cryptography

OpenFileStorage(): FileStorage

OpenFileWriteOnly(): ReadWriteAccessor

open()

alt

[KeySlotMapping exists in Manifest and keySlotUsage = encryption]

ref

Initialize Cipher

WriteText()

alt

[KeySlotMapping exists in Manifest and keySlotUsage = encryption]

ProcessBlock()

write()

**Figure 11.9:** **Scenario: Store Data to File Successfully**

Figure [11.9](#_bookmark280) shows a scenario for storing data to a file successfully. The [Adaptive](#_bookmark24) [Application](#_bookmark24) needs to open a [FileStorage](#_bookmark130) by calling [OpenFileStorage()](#_bookmark129). Afterwards, the [Adaptive Application](#_bookmark24) needs to open an individual file for writing by calling [OpenFileWriteOnly()](#_bookmark133) or [OpenFileReadWrite()](#_bookmark132) depending if the [Adaptive Applica-](#_bookmark24) [tion](#_bookmark24) needs to read data from the file as well. Then, the [Adaptive Application](#_bookmark24) can store data via the methods provided by [ReadWriteAccessor](#_bookmark135) either as binary data or text data.

If enabled in the Manifest, [Persistency](#_bookmark124) stores redundant data to detect and cor- rect errors when reading from a file later. If the file or [FileStorage](#_bookmark130) is configured to use encryption, data will be encrypted before it is written to the underlying storage.

* + - 1. **Store Persistent Data with Unique ID**
         1. **Scenario: Store Data with Unique ID Successfully**



:Adaptive Application



:Operating System

«aapFunctionalClust...

:Persistency

«aapFunctionalClust...

:Cryptography



OpenKeyValueStorage(): KeyValueStorage

SetValue()

SyncToStorage()

open()

alt

[KeySlotMapping exists in Manifest and keySlotUsage = encryption]

ref

Initialize Cipher

ProcessBlock()

write()

**Figure 11.10:** **Scenario: Store Persistent Data with a Unique ID Successfully**

Figure [11.10](#_bookmark282) shows a scenario for storing data associated with a unique identifier (key) successfully. The [Adaptive Application](#_bookmark24) needs to open a [KeyValueStorage](#_bookmark141) by calling [OpenKeyValueStorage()](#_bookmark140). Then, the [Adaptive Application](#_bookmark24) can store data associated to a key by calling [SetValue()](#_bookmark143). The data is updated in memory by calling [SetValue()](#_bookmark143) but not written to persistent storage. The [Adaptive Application](#_bookmark24) needs to call [SyncToStorage()](#_bookmark144) to write one or more such changes to persistent storage.

If enabled in the Manifest, [Persistency](#_bookmark124) stores redundant data to detect and cor- rect errors when reading from a [KeyValueStorage](#_bookmark141) later. If the key or [KeyValueStorage](#_bookmark141) is configured to use encryption, data will be encrypted before it is written to the under- lying storage.

* + - 1. **Get Storage Size**
         1. **Scenario: File Store**

getSize()

GetCurrentFileStorageSize()

«aapFunctionalClust...

:Persistency



:Adaptive Application :Operating System

**Figure 11.11:** **Scenario for determining the size of a FileStorage**

Figure [11.11](#_bookmark284) shows the scenario for monitoring the storage space of a [FileStorage](#_bookmark130). The [Adaptive Application](#_bookmark24) needs to call [GetCurrentFileStorageSize()](#_bookmark128) to determine the current size.

* + - * 1. **Scenario: Key Value Store**



:Adaptive Application :Operating System

«aapFunctionalClust...

:Persistency

GetCurrentKeyValueStorageSize()

getSize()

**Figure 11.12:** **Scenario for determining the size of a KeyValueStorage**

Figure [11.12](#_bookmark286) shows the scenario for monitoring the storage space of a [KeyValueStor-](#_bookmark141) [age](#_bookmark141). The [Adaptive Application](#_bookmark24) needs to call [GetCurrentKeyValueStorageSize()](#_bookmark139) to determine the current size.

* 1. **Security**
     1. **Firewall**
        1. **Switch Firewall State**
           1. **Scenario: Switch Firewall State Successfully**



:Adaptive Application :Operating System

**Figure 11.13:** **Scenario: Switch Firewall State Successfully**

ewallRule]

applyRule()

loop [foreach Fir

SwitchFirewallState()

«aapFunctionalCl...

:Firewall

Figure [11.13](#_bookmark290) shows the scenario for successfully switching the state of the [Firewall](#_bookmark182). The [Adaptive Application](#_bookmark24) triggers the state switch by calling [SwitchFirewallState](#_bookmark183) [()](#_bookmark183). The [Firewall](#_bookmark182) will then apply the FirewallRules configured for the request state

in an implementation-specific way (e.g., using tools provided with the TCP/IP stack of the [Operating System](#_bookmark28)).

* + - 1. **Report Security Event**
         1. **Scenario: Report Security Event Successfully**



:Operating System

«aapFunctionalCl...

:Firewall



«aapFunctionalCluster»

:Adaptive Intrusion Detection System Manager

packetBlocked() handlePacketBlocked()

raiseSecurityEvent() ReportEvent()

**Figure 11.14:** **Scenario: Report Security Event Successfully**

Figure [11.14](#_bookmark292) shows the scenario for successfully reporting a Security Event to [Adaptive Intrusion Detection System Manager](#_bookmark178). The [Operating System](#_bookmark28) (or another component that implements the actual firewall) reports that a packet has been blocked by a specific rule to the [Firewall](#_bookmark182). If a Security Event has been configured for that rule in the Manifest, the [Firewall](#_bookmark182) will create a correspond- ing Security Event and report it to [Adaptive Intrusion Detection System](#_bookmark178) [Manager](#_bookmark178) by calling [ReportEvent()](#_bookmark180). [Adaptive Intrusion Detection System](#_bookmark178) [Manager](#_bookmark178) will then handle the Security Event accordingly.

1. **Deployment View**

This chapter provides an overview of exemplary deployment scenarios for an AU- TOSAR Adaptive Platform. Since the AUTOSAR Adaptive Platform is highly config- urable in its deployment, this section rather provides constraints on supported deploy- ments and a selection of relevant deployment scenarios.

## Vehicle Software Deployment



«device» gateway: Machine

«executionEnvironment»

:Adaptive Runtime

UCM Master

«manifest»

«manifest»

«manifest»

«device» ecu1: ECU

«device» machine1: Machine

«manifest»

UCM Subordinate

«executa... Update and Configuration Management

«executa...

OTA Client Application

OTA Client

«device» Vehicle

:Diagnostic Client

:Backend System

«executionEnvironment»

:Adaptive Runtime

«executa... Update and Configuration Management

«executionEnviron...

:Classic Platform

**Figure 12.1: Exemplary vehicle software update scenario**

[Update and Configuration Management](#_bookmark200) allows to install and update software on the AUTOSAR Adaptive Platform and AUTOSAR Classic Platform. For the AU- TOSAR Adaptive Platform, [Update and Configuration Management](#_bookmark200) also al- lows to remove software. The software packages can be received either from a Di- agnostic Client or from a specific Backend System for over-the-air updates. In a vehicle, one [Adaptive Application](#_bookmark24) takes the role of a master that controls the update process in the vehicle and distributes individual software packages to the Ma- chines and ECUs within a vehicle.

## Deployment of Software Packages on a Machine

**Legend**

user

platformVendor machineVendor



«manifest»

«manifest»

«manifest»

«aapFunctionalCl...

State Management daemon-based

Watchdog

Operating System

«manifest»

«manifest»

«manifest»

*tags*

Multiplicity = 1

SoftwareCluster.category = PLATFORM\_CORE

«software package» Platform Core Package

«aapFunctionalCl... Execution Management

daemon-based

«aapFunctionalClust...

Platform Health Management

daemon-based

«aapFunctionalCluster» Update and Configuration Management

daemon-based

«manifest»

«manifest»

*tags*

Multiplicity = \* SoftwareCluster.category = PLATFORM

«software package» Platform Package

«aapFunctionalClust...

Cryptography daemon-based

«aapFunctionalCluster»

Diagnostic Management daemon-based

Platform

*tags*

Multiplicity = \*

SoftwareCluster.category = APPLICATION\_LAYER

«software package» Application Package

«device» Machine

Bootloader / Hypervisor

**Figure 12.2:** **Exemplary deployment of Software Packages on a Machine**

[Update and Configuration Management](#_bookmark200) manages the software installed on a Machine using one or more Software Packages. The configuration of the contents of a Software Package is done in the Manifest by means of a Software Clus- ter. Software Cluster defines the enumeration attribute category with the val- ues APPLICATION, PLATFORM, and PLATFORM\_CORE. Some rules apply on the use of the Software Cluster category and corresponding Software Packages which are outlined in the following.

For a Machine there shall exist exactly one Software Cluster with category PLATFORM\_CORE. The corresponding Platform Core Package needs to include all required core components provided by the Platform Vendor and Machine Vendor, for example the operating system, device drivers, and the required Func- tional Clusters of the Adaptive Platform (see Figure [12.2](#_bookmark296)). It is assumed that at least the Functional Clusters [Execution Management](#_bookmark61), [State Management](#_bookmark76), and [Update and Configuration Management](#_bookmark200) are part of the Platform Core Package since they are required to install any additional Software Packages. How- ever, the concrete extend of a Platform Core Package is vendor and/or project specific. A Platform Core Package cannot be removed. The Software Clus- ter with category PLATFORM\_CORE needs to be self-contained and therefore shall not have dependencies to other Software Clusters.

Optional Functional Clusters of the Adaptive Platform may be distributed as part of any number of additional Software Clusters with category PLATFORM within Platform Packages. Such Software Clusters may depend on the the

Software Cluster with category PLATFORM\_CORE (see Figure [12.2](#_bookmark296)). Platform Packages may be installed and removed as needed.

Application components should be distributed as part of additional Software Clus- ters with category APPLICATION within Application Packages. Such Soft- ware Clusters may depend on the on Software Clusters with any kind of cat- egory (see Figure [12.2](#_bookmark296)). Application Packages may be installed and removed as needed.

The Platform Core Package and Platform Packages shall include all required Functional Cluster daemons and their respective configuration (e.g., startup con- figuration). This frees applications (distributed as Application Packages) from tak- ing care of the configuration of the platform.

# Cross-cutting Concepts

This section provides an overview of cross-cutting concepts and patterns used in the AUTOSAR Adaptive Platform.

## Overview of Platform Entities

The AUTOSAR Adaptive Platform defines design entities that several Functional Clusters depend on. Figure [13.1](#_bookmark299) provides an overview of these entities, their logical relationships, and the Functional Clusters that depend on them. For the sake of brevity, this overview uses simplifications and abstractions over the actual specifica- tions in the [[9](#_bookmark331), manifest specification].



0..\*

2..\*

+state

0..\*

+/requiredCluster 0..\*

+executionDependecy 0..\*

1 +executable

+/checkpoints

1..\*

Adaptive Application

«aapFunctionalCluster»

Platform Health Management daemon-based

Supervision Checkpoint

*PhmSupervision*

1

+process

Process

Executable

0..\*

* options
* schedulingPriority
* diagnosticConfig [0..1]
* version

1

Startup Configuration

Function Group State

Function Group

Software Cluster

Software Package

«aapFunctionalCluster»

Execution Management daemon-based

«aapFunctionalCluster»

State Management daemon-based

«aapFunctionalCluster»

Update and Configuration Management daemon-based

**Figure 13.1:** **Overview of platform entities and their logical relationships**

A Software Package is a digitally signed package that can be installed/uninstalled via [Update and Configuration Management](#_bookmark200). A Software Package contains exactly one Software Cluster (see Section [13.4](#_bookmark302) for details). A Software Clus- ter refers to a set of Executables (and other files). The corresponding executable file then holds the executable code for the Machine that the AUTOSAR Adaptive Plat- form runs on.

Additionally, a Software Cluster configuration collects a set of Processes (cf. Section [13.4](#_bookmark302)) and related entities. A Process refers to an Executable and provides different Startup Configuration values, for example parameters, a scheduling

priority, and resource constraints. A Startup Configuration of a Process ap- plies to one or more Function Group States. Function Group States belong to a Function Group.

During runtime, [State Management](#_bookmark76) requests to enter a Function Group State from [Execution Management](#_bookmark61). [Execution Management](#_bookmark61) then terminates and starts the Processes accordingly using the underlying [Operating System](#_bookmark28).

For safety-critical systems, [Platform Health Management](#_bookmark186) performs supervision of Processes according to rules (logical sequence, deadlines) defined in PhmSuper- visions. A PhmSupervision refers to a number of Supervision Checkpoints. During runtime, a process reports whenever it has reached such a checkpoint in its control flow.

## Function Group

A Function Group is (logically) made up a set of modeled Processes that provide a certain functionality. For example, a Function Group could be an application, or a service. A special Function Group is the Machine State that groups the Process of the AUTOSAR Adaptive Platform itself. A Function Group contains a set of Function Group States.

## Function Group State

A Function Group State defines which Processes of a Function Group with what configuration parameters shall be running or not. The Machine State (that refers to the Processes of the AUTOSAR Adaptive Platform itself) defines at least the following Function Group States: Off, Startup, Shutdown, and Restart.

## Software Cluster

A Software Cluster configuration refers to a set of modeled Processes. Those Processes are (transitively) used by one or more Function Group(s). Hereby, a Function Group and its associated entities shall be part of only one Software Cluster. In other words, Function Groups that span several Software Clus- ters are invalid. A Software Cluster is packaged into one Software Package

- the atomic installable/updateable unit managed by Update and Configuration Management. A Software Cluster may depend on other Software Clusters. Such dependencies are expressed by version constraints. A Software Cluster may also specify structural dependencies to Sub Software Clusters in order to build larger installable units. The top of such a structural dependency hierarchy is called a Root Software Cluster. Please note that a Software Cluster is only used to configure deployment aspects. A Software Cluster is not a runtime entity.

A Root Software Cluster may specify a diagnostic configuration, in particular, a diagnostic address. In contrast, a Sub Software Cluster may depend on a diag- nostic configuration of its Root Software Cluster. The diagnostic configuration applies to Processes that are (transitively) contained in a Root Software Clus- ter and its Sub Software Cluster(s). That means, at runtime, any diagnostic event produced by those Processes will be associated with the diagnostic address. Please refer to Section [13.11](#_bookmark311) for further details on the diagnostic deployment.

An exemplary Software Cluster during application design is shown in Figure [13.2](#_bookmark303). The application Software Cluster(s) are modeled/configured in the same way as the platform Software Cluster by defining Function Groups with Function Group States and associating StartupConfigurations of Processes to them.

A Software Cluster serves as a grouping entity during application design. As a result, entities within a Software Cluster, in particular the Function Groups, do not need to have a unique (simple) name within the overall model because their path is still unique. This allows to design Software Clusters independently, for example, by external suppliers.

a: Software Cluster

e1: Executable

executionManifest: Manifest

p1: Process

cfg1: Startup Configuration

fg1: Function Group

off: Function Group normal: Function State Group State

degraded: Function Group State

b: Software Cluster

e2: Executable

executionManifest: Manifest

p2: Process

cfg2: Startup Configuration

fg2: Function Group

off: Function Group normal: Function State Group State



platform: Software Cluster

:Manifest

EM: Process

:Startup Configuration

**Figure 13.2:** **Exemplary Software Cluster during application design**

From such a standardized model, an equivalent implementation-specific configura- tion for [Execution Management](#_bookmark61) is derived that is used during runtime (see Figure [13.3](#_bookmark304)). That configuration advises [Execution Management](#_bookmark61) to start and configure processes accordingly when a Function Group State is requested. Hereby, [Ex-](#_bookmark61) [ecution Management](#_bookmark61) (logically) merges configurations contributed by all installed Software Packages. Other Functional Clusters that depend on configuration provided in the Manifests merge the configurations contributed by all installed Soft- ware Packages in the same way. Please also note that there is no corresponding runtime entity for a Software Cluster (see Figure [13.3](#_bookmark304)).



State

:Execution Management

Platform EM Configuration

fg2 EM Configuration

fg1 EM Configuration

normal: Function Group State

off: Function Group

normal: Function Group State

off: Function Group State

fg2: Function Group

fg1: Function Group

cfg1: Startup Configuration

cfg2: Startup Configuration

p2: Process

p1: Process

e2: Executable

e1: Executable

degraded: Function Group State

**Figure 13.3:** **Impact of exemplary Software Cluster during runtime**

All Processes related to the Functional Clusters of the AUTOSAR Adaptive Platform should be referenced only in Software Clusters of category PLAT- FORM\_CORE or PLATFORM. This allows for platform-independent development of Software Clusters of category APPLICATION\_LAYER.

In case a Functional Cluster may need multiple logical instances (for example, Diagnostic Management has a logical instance per diagnostic address), an imple- mentation of the Functional Cluster should still use a single physical (daemon) process.

An AUTOSAR Adaptive Platform vendor may deviate from this design guide but should provide additional countermeasures to keep [Adaptive Application](#_bookmark24)s portable.

## Machine

The AUTOSAR Adaptive Platform regards hardware it runs on as a Machine. The ra- tionale behind that is to achieve a consistent platform view regardless of any virtualiza- tion technology which might be used. The Machine might be a real physical machine, a fully-virtualized machine, a para-virtualized OS, an OS-level-virtualized container or any other virtualized environment.

On hardware, there can be one or more Machine, and only a single instance of AU- TOSAR Adaptive Platform runs on a machine. It is generally assumed that this hard- ware includes a single chip, hosting a single or multiple Machines. However, it is also possible that multiple chips form a single Machine if the AUTOSAR Adaptive Platform implementation allows it.

## Manifest

A Manifest represents a piece of AUTOSAR model description that is created to support the configuration of an AUTOSAR Adaptive Platform product and which is up- loaded to the AUTOSAR Adaptive Platform product, potentially in combination with other artifacts (like binary files) that contain executable code to which the Manifest applies. Please note that a typical [Adaptive Application](#_bookmark24) will make use of several distinct but interrelated Manifests. Hereby, the individual Manifests contribute in- formation to the complete application model. For example, each Software Cluster may contribute a self-contained set of Manifests to configure its functionality.

The usage of a Manifest is limited to the AUTOSAR Adaptive Platform. This does not mean, however, that all ARXML produced in a development project that targets the AUTOSAR Adaptive Platform is automatically considered a Manifest. In fact, the AUTOSAR Adaptive Platform is usually not exclusively used in a vehicle project. A typical vehicle will most likely be also equipped with a number of ECUs developed on the AUTOSAR Classic Platform and the system design for the entire vehicle will, therefore, have to cover both, ECUs built on top of the AUTOSAR Classic Platform and Machines created on top of the AUTOSAR Adaptive Platform.

In principle, the term Manifest could be defined such that there is conceptually just one "Manifest" and every deployment aspect would be handled in this context. This does not seem appropriate because it became apparent that Manifest-related model- elements exist that are relevant in entirely different phases of a typical development project.

This aspect is taken as the main motivation that next to the application design it is necessary to subdivide the definition of the term Manifest in three different partitions:

**Application Design** This kind of description specifies all design-related aspects that apply to the creation of application software for the AUTOSAR Adaptive Platform. It is not necessarily required to be deployed to the adaptive platform machine, but the application design aids the definition of the deployment of application software in the

Execution Manifest and Service Instance Manifest. See Section [13.7](#_bookmark307) for details.

**Execution Manifest** This kind of Manifest is used to specify the deployment-related information of applications running on the AUTOSAR Adaptive Platform. An Execu- tion Manifest is bundled with the actual executable code to support the integration of the executable code onto the machine. See Section [13.8](#_bookmark308) for details.

**Service Instance Manifest** This kind of Manifest is used to specify how service- oriented communication is configured in terms of the requirements of the underlying transport protocols. A Service Instance Manifest is bundled with the actual executable code that implements the respective usage of service-oriented communi- cation. See Section [13.9](#_bookmark309) for details.

**Machine Manifest** This kind of Manifest is supposed to describe deployment-related content that applies to the configuration of just the underlying machine (i.e. without any applications running on the machine) that runs an AUTOSAR Adaptive Platform. A Machine Manifest is bundled with the software taken to establish an instance of the AUTOSAR Adaptive Platform. See Section [13.10](#_bookmark310) for details.

The temporal division between the definition (and usage) of different kinds of Mani- fest leads to the conclusion that in most cases different physical files will be used to store the content of the three kinds of Manifest. In addition to the Application Design and the different kinds of Manifest, the AUTOSAR Methodology supports a Sys- tem Design with the possibility to describe Software Components of both AUTOSAR Platforms that will be used in a System in one single model. The Software Compo- nents of the different AUTOSAR platforms may communicate in a service-oriented way with each other. But it is also possible to describe a mapping of Signals to Services to create a bridge between the service-oriented communication and the signal-based communication.

## Application Design

The application design describes all design-related modeling that applies to the cre- ation of application software for the AUTOSAR AP. Application Design focuses on the following aspects:

* Data types used to classify information for the software design and implementa- tion
* Service interfaces as the pivotal element for service-oriented communication
* Definition how service-oriented communication is accessible by the application
* Persistency Interfaces as the pivotal element to access persistent data and files
* Definition how persistent storage is accessible by the application
* Definition how files are accessible by the application
* Definition how crypto software is accessible by the application
* Definition how the Platform Health Management is accessible by the application
* Definition how Time Bases are accessible by the application
* Serialization properties to define the characteristics of how data is serialized for the transport on the network
* Description of client and server capabilities
* Grouping of applications in order to ease the deployment of software.

The artifacts defined in the application design are independent of a specific deployment of the application software and thus ease the reuse of application implementations for different deployment scenarios.

## Execution Manifest

The purpose of the execution manifest is to provide information that is needed for the actual deployment of an application onto the AUTOSAR AP. The general idea is to keep the application software code as independent as possible from the deployment scenario to increase the odds that the application software can be reused in different deployment scenarios. With the execution manifest the instantiation of applications is controlled, thus it is possible to

* instantiate the same application software several times on the same machine, or to
* deploy the application software to several machines and instantiate the applica- tion software per machine.

The Execution manifest focuses on the following aspects:

* Startup configuration to define how the application instance shall be started. The startup includes the definition of startup options and access roles. Each startup may be dependent on machines states and/or function group states.
* Resource Management, in particular resource group assignments.

## Service Instance Manifest

The implementation of service-oriented communication on the network requires con- figuration which is specific to the used communication technology (e.g. SOME/IP). Since the communication infrastructure shall behave the same on the provider and the requesters of a service, the implementation of the service shall be compatible on both sides.

The Service Instance Manifest focuses on the following aspects:

* Service interface deployment to define how a service shall be represented on the specific communication technology.
* Service instance deployment to define for specific provided and required service instances the required credentials for the communication technology.
* The configuration of E2E protection
* The configuration of Security protection
* The configuration of Log and Trace

## Machine Manifest

The machine manifest allows to configure the actual adaptive platform instance running on specific hardware (machine).

The Machine Manifest focuses on the following aspects:

* Configuration of the network connection and defining the basic credentials for the network technology (e.g. for Ethernet this involves setting of a static IP address or the definition of DHCP).
* Configuration of the service discovery technology (e.g. for SOME/IP this involves the definition of the IP port and IP multi-cast address to be used).
* Definition of the used machine states.
* Definition of the used function groups.
* Configuration of the adaptive platform functional cluster implementations (e.g. the operating system provides a list of OS users with specific rights).
* The configuration of the Crypto platform Module.
* The configuration of Platform Health Management.
* The configuration of Time Synchronization.
* Documentation of available hardware resources (e.g. how much RAM is avail- able; how many processor cores are available).

## Diagnostics deployment

A diagnostic configuration, in particular a diagnostic address, may only be assigned to a Root Software Cluster in the Manifest. Nevertheless, the mapped Diag- nosticContributionSet(s) may be distributed across several Software Clus- ters. This concept provides a lot of flexibility in assignment of a single diagnostic address to Software Clusters. For example, in one extreme this allows to use a single diagnostic address for the whole Machine (see Figure [13.4](#_bookmark312)), in another extreme

a single diagnostic address per Software Cluster could be used (see Figure [13.5](#_bookmark313)). Deployment scenarios in between those extremes are also possible.

Shared diagnostic address 0x2



Shared diagnostic address 0x1234

- category = PLATFORM

* category = PLATFORM\_CORE
* diagnosticAddress = 0x1234

Platform: Software Cluster

Platform Core: Software Cluster

- category = APPLICATION\_LAYER

- category = APPLICATION\_LAYER

- DID 0x2314

- DID 0x4321

app2Sub2: Software Cluster

app2Sub1: Software Cluster

shared: DiagnosticContributionSet

- category = APPLICATION\_LAYER

- category = APPLICATION\_LAYER

app2Root: Software Cluster

app1: Software Cluster



**Figure 13.4:** **Example defining a single diagnostic address for the whole Machine**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Diagnostic address 0x1 | | | | |
|  | app1: Software Cluster | | |  |
|  | * category = APPLICATION\_LAYER * diagnosticAddress = 0x1 | | |  |
|  | |  | | |
|  | :DiagnosticContributionSet | | |  |
|  | - DID 0x1234  - DID 0x2314 | | |  |
|  | | | | |
|  | | | | |
| Dinagnostic address 0x3 | | | | |
|  | Platform Core: Software Cluster | | |  |
|  | * category = PLATFORM\_CORE * diagnosticAddress = 0x3 | | |  |
|  | |  | | |
|  | :DiagnosticContributionSet | | |  |
|  | - DID 0x1234 | | |  |
|  |  | |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| app2Root: Software Cluster | | | | |
| * category = APPLICATION\_LAYER * diagnosticAddress = 0x2 | | | | |
|  |  | | |  |
| app2Sub1: Software Cluster | |  | app2Sub2: Software Cluster | |
| - category = APPLICATION\_LAYER | | - category = APPLICATION\_LAYER | |



**Figure 13.5:** **Example using one diagnostic address for each Software Cluster**

In the case of a distributed DiagnosticContributionSet, each Software Clus- ter shall include its related diagnostic configuration objects (for example, the Data Identifier configuration). The merge of such a split DiagnosticContribution- Set is done internally by the AUTOSAR Adaptive Platform (e.g. during installation or during start up the Diagnostic Management daemon).



- DID 0x1234

:DiagnosticContributionSet

* category = PLATFORM
* diagnosticAddress = 0x4

Platform: Software Cluster

Diagnostic address 0x4

## Error Handling

Proper handling of errors during runtime is an important aspect to build safe and se- cure systems. The AUTOSAR Adaptive Platform does provide means for raising and handling of such errors on different levels in the platform.

[Platform Health Management](#_bookmark186) detects errors (errors in the logical control flow, missed deadlines, and missed liveness reporting) at the level of Processes and performs recovery actions (for example, degradation) according to rules defined in the Manifest. [Execution Management](#_bookmark61) detects unexpected termination of Pro- cesses and reports to [State Management](#_bookmark76) for handling of such errors.

During execution of a Process of an [Adaptive Application](#_bookmark24), different abnormal conditions might be detected and need to be handled and/or reported. The follow- ing types of unsuccessful operations are distinguished within the AUTOSAR Adaptive Platform:

* An Error is the inability of an AUTOSAR Runtime for Adaptive Applications API function to fulfill its specified purpose. An Error it is often a consequence of invalid and/or unexpected input data. An Error is considered to be recoverable and therefore shall be handled by applications.
* A Violation is the consequence of failed pre- or post-conditions of internal state of the AUTOSAR Runtime for Adaptive Applications. A Violation is con- sidered to be non-recoverable.
* A Corruption is the consequence of the corruption of a system resource, e.g. stack or heap overflow, or a hardware memory flaw (for example, a detected bit flip). A Corruption is considered to be non-recoverable.
* A failed default allocation is the inability of the AUTOSAR Runtime for Adaptive Applications’s default memory allocation mechanism to satisfy an allo- cation request (for example, there is not enough free memory available).

It is expected that a Violation or Corruption will not be experienced by a user of the AUTOSAR Adaptive Platform (i.e. an application developer), unless there is some- thing seriously wrong in the overall system. For example, faulty hardware may lead to a Corruption. A Violation may occur if basic assumptions about resource require- ments are violated, or the user runs the AUTOSAR Runtime for Adaptive Applications in a configuration that is not supported by its vendor.

## Trusted Platform

To guarantee the correct function of the system, it is crucial to ensure that the code ex- ecuted on the AUTOSAR Adaptive Platform is unaltered (integrity) and has legitimate origin (authenticity). Keeping this property allows the integrator to build a Trusted Platform. A key property of a system that implements a Trusted Platform is a Trust Anchor (also called Root of Trust). A Trust Anchor is often realized

as a public key that is stored in a secure environment, e.g. in non-modifiable persistent memory or in an Hardware Security Module. A system designer is responsible to ensure that the system starts beginning with a Trust Anchor and that the chain of trust is kept until the [Execution Management](#_bookmark61) is launched. Depending on the mech- anism that is chosen by the system designer to establish the chain of trust, the integrity and authenticity of the entire system (including all executables) may be checked during system start-up. Alternatively, the system designer might only ensure that the already executed software has been checked regarding integrity and authenticity and [Execu-](#_bookmark61) [tion Management](#_bookmark61) takes over responsibility on continuing the chain of trust when it takes over control of the system. In the latter case, the system integrator is responsible to ensure that the [Execution Management](#_bookmark61) is configured accordingly.

Passing trust requires that a trusted entity checks (using trusted functionality) that the entity down the chain is authentic. The Trust Anchor (the first entity in the chain) is authentic by definition. An example of such a chain of trust could look like this: The Trust Anchor authenticates the bootloader before the bootloader is being started. In each subsequent step in the boot process, the to-be-started executable shall be au- thenticated first, for example by the executable started previously or by some external entity like an Hardware Security Module. After the relevant parts of the [Operat-](#_bookmark28) [ing System](#_bookmark28) have been authentically started, it shall launch [Execution Manage-](#_bookmark61) [ment](#_bookmark61) as one of its first processes in the same manner passing trust to the AUTOSAR Adaptive Platform. Then, [Execution Management](#_bookmark61) takes over the responsibility of authenticating [Adaptive Application](#_bookmark24)s before launching them.

As stated above, if authenticity is not checked by the functionality of the Trust An- chor itself, which is authentic by definition, the functionality that shall be applied to verify authenticity of an executable has to be authenticated as well before it is applied. For instance, if the Crypto Functional Cluster shall be used to verify authentic- ity of executables, the Crypto Functional Cluster itself shall be authenticated by some trusted entity before it is used.

## Secure Communication

AUTOSAR supports different protocols that provide communication security over a net- work. Integrity of messages can be ensured by the end-to-end protection offered by the [[10](#_bookmark332), AUTOSAR E2E library]. End-to-end protection assumes that safety- and security- related data exchange shall be protected at runtime against the effects of faults within the communication link. Such faults include random hardware faults (e.g. corrupt regis- ters of a transceiver), interference (e.g. electromagnetic interference), and systematic faults in the communication stack. The configuration of end-to-end-protection is done via Service Instance Manifest on level of Service events, methods, and fields (notifier, get, and set methods). Confidentiality and authenticity of messages can be ensured by dedicated configurations for the individual transport protocols (e.g. TLS, SecOC) in the Service Instance Manifest on level of Service events, meth- ods, and fields (notifier, get, and set methods).

# Risks and Technical Debt

This chapter lists and rates risks associated with the overall architecture of the AU- TOSAR Adaptive Platform in Section [14.1](#_bookmark318). These risks usually might cause that some of the quality attributes of the AUTOSAR Adaptive Platform are not (fully) meat. Sec- tion [14.2](#_bookmark322) lists technical debt of the AUTOSAR Adaptive Platform that may impact its maintainability.

## Risks

### Risk Assessment

This document categorizes risks according to their severity. The severity is a function of the probability and the impact of a risk. The probabilities are categorized as follows:

* **very low** - probability is less than 1 thousandth
* **low** - probability is between 1 thousandth and 1 percent
* **medium** - probability is between 1 percent and 10 percent
* **high** - probability is between 10 percent and 50 percent
* **very high** - probability is more than 50 percent The impact of a risk is categorized as follows:
* **very low** - at most one quality scenario will take additional significant effort to be satisfied
* **low** - more than one quality scenario will take additional significant effort to be satisfied
* **medium** - at most one quality scenario is not satisfied with small gaps
* **high** - at most one quality scenario is not satisfied with big gaps
* **very high** - more than one quality scenario is not satisfied with big gaps The final severity of a risk is then calculated according to table [14.1](#_bookmark320).

**Probability**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Impact** | very low | low | medium | high | very high |
| very low | low (1) | low (2) | low (3) | medium (4) | medium (5) |
| low | low (2) | medium (4) | medium (6) | high (8) | high (10) |
| medium | low (3) | medium (6) | high (9) | high (12) | high (15) |
| high | medium (4) | high (8) | high (12) | extreme (16) | extreme (20) |
| very high | medium (5) | high (10) | high (15) | extreme (20) | extreme (25) |

**Table 14.1: Risk Severity Matrix**

### Risk List

No architectural risks were identified yet.

## Technical Debt

No technical debt has been identified yet.

# References

1. ISO 42010:2011 – Systems and software engineering – Architecture description [http://www.iso.org](http://www.iso.org/)
2. Explanation of Adaptive and Classic Platform Software Architectural Decisions AUTOSAR\_EXP\_SWArchitecturalDecisions
3. Glossary AUTOSAR\_TR\_Glossary
4. Main Requirements AUTOSAR\_RS\_Main
5. General Requirements specific to Adaptive Platform AUTOSAR\_RS\_General
6. ATAMSM: Method for Architecture Evaluation [https://resources.sei.cmu.edu/asset\_files/TechnicalReport/2000\_005\_001](https://resources.sei.cmu.edu/asset_files/TechnicalReport/2000\hskip%200em%7B%7D_005\hskip%200em%7B%7D_001\hskip%200em%7B%7D_13706.pdf)

[\_13706.pdf](https://resources.sei.cmu.edu/asset_files/TechnicalReport/2000\hskip%200em%7B%7D_005\hskip%200em%7B%7D_001\hskip%200em%7B%7D_13706.pdf)

1. Agile Software Development: Principles, Patterns, and Practices
2. Guide to the Software Engineering Body of Knowledge, Version 3.0 [www.swebok.org](http://www.swebok.org/)
3. Specification of Manifest AUTOSAR\_TPS\_ManifestSpecification
4. Specification of SW-C End-to-End Communication Protection Library AUTOSAR\_SWS\_E2ELibrary