Document Title	Specification of Identity and Access Management
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	900

Document Status	Final
Part of AUTOSAR Standard	Adaptive Platform
Part of Standard Release	19-03

Document Change History			
Date	**Release**	**Changed by**	**Description**
2019-03-29	19-03	AUTOSAR Release Management	• Reworked chapter 7 to incorporate new concept of `Grants` • Incorporation of several bug tickets
2018-10-31	18-10	AUTOSAR Release Management	• Reworked functional specification • Removed API specification for general rework
2018-03-29	18-03	AUTOSAR Release Management	• Initial release

Disclaimer

Table of Contents

1 Introduction and functional overview

This specification describes the functionality, API and the configuration for the Identity and Access Management functional cluster of the AUTOSAR Adaptive Platform.
The Identity and Access Management offers applications a standardized interface to access management operations.

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the Identity and Access Management module that are not included in the AUTOSAR glossary [1].

Abbreviation / Acronym:	Description:
PDP	Policy Decision Point
PEP	Policy Enforcement Point
IPC	Inter-Process Communication

3 Related documentation

3.1 Input documents & related standards and norms

[1] Glossary
AUTOSAR_TR_Glossary

[2] Requirements on Identity and Access Management
AUTOSAR_RS_IdentityAndAccessManagement

3.2 Related standards and norms

See chapter 3.1.

3.3 Related specification

See chapter 3.1.

4 Constraints and assumptions

4.1 Known Limitations

- A detailed API will be added probably in Release R19-11.
- Currently limited to ara::com
- For other Functional Clusters, implementation on Policy Enforcement Points are envisaged for the next release (R19-11).

4.2 Assumptions

The integrator can configure an authentic channel between `Policy Decision Points` and `Policy Enforcement Points`. This could be done through the operating system's access rights for example.

5 Dependencies to other modules

The implementation of a `Policy Enforcement Point` in a `Functional Cluster` is defined in the corresponding software specification.

The following implementations are defined:

- Communication Management (Section 7.5 Security)

6 Requirements Tracing

The following tables reference the requirements specified in [2] and links to the fulfillment of these. Please note that if column "Satisfied by" is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[RS_IAM_00002]	Enforcement of access control shall happen within `Adaptive Platform Foundation`	[SWS_IAM_01001] [SWS_IAM_02009]
[RS_IAM_00004]	Circumvention of AUTOSAR PEP interfaces by Applications shall be prevented.	[SWS_IAM_02010]

Requirement	Description	Satisfied by
[RS_IAM_00005]	`Adaptive Platform Foundation` shall enforce that only Applications that are configured accordingly are able to gain information about the permissions of other applications	[SWS_IAM_02000]
[RS_IAM_00008]	Access shall be denied by the PEP if the corresponding PDP is not available	[SWS_IAM_02004]
[RS_IAM_00009]	An `Adaptive Application` may provide access control decisions	[SWS_IAM_02001] [SWS_IAM_02005] [SWS_IAM_02006] [SWS_IAM_02007] [SWS_IAM_02008] [SWS_IAM_02011]
[RS_IAM_00010]	Adaptive applications shall only be able to use AUTOSAR Resources when authorized	[SWS_IAM_01002] [SWS_IAM_02001] [SWS_IAM_02003]
[RS_IAM_00011]	Policies shall be enforced by the local `Adaptive Platform Foundation`	[SWS_IAM_02003] [SWS_IAM_02005] [SWS_IAM_02006] [SWS_IAM_02007] [SWS_IAM_02008] [SWS_IAM_02011] [SWS_IAM_02012] [SWS_IAM_02013] [SWS_IAM_02014] [SWS_IAM_02015]
[RS_IAM_00012]	No description	[SWS_IAM_02011] [SWS_IAM_02012] [SWS_IAM_02013] [SWS_IAM_02014] [SWS_IAM_02015]
[RS_IAM_00014]	Unique `Adaptive Application` ID	[SWS_IAM_02013]
[RS_IAM_00015]	No description	[SWS_IAM_02013]

7 Functional specification

The functional specifications provides a technical overview to elaborate the generic approach to modeling and implementing Identity and Access Management features in the Adaptive Platform.

The technical concept of IAM shall be discussed from a modeling and a runtime perspective. The modeling perspective shall elaborate the concept from the view of an application designer and an integrator. The runtime perspective shall outline the integration of the concept into the Adaptive Platform functional clusters.

7.1 Terms

Before discussing concept of Identity and Access Management the employed terminology shall be elaborated.

- **Capability -** A capability is a property of an Adaptive Application. Access to an AUTOSAR resource (e.g. `ServiceInterface` and its members `Method`, `Event` and `Field`) is granted only if a requesting Adaptive Application pos-

sesses all Capabilities that are mandatory for that specific resource. Capabilities are assigned to Adaptive Applications within their Application Manifest by means of `ComSpecs` (e.g. `ClientComSpec`) and `GrantDesigns` (e.g. `ComFieldGrantDesign`).

- **Grant -** The integrator acknowledges an Adaptive Application's capabilities by transferring `GrantDesigns` to a `Grant` in deployment. `Grant` elements may be processed into access control lists for the PDP implementation.
- **Process -** A `Process` is the meta model's runtime instance of an Adaptive Application and represents its runtime identity. A `Process` may be identified during runtime by an uniquely assigned identifier (e.g. a Unix user).
- **-**
- **Policy Enforcement Point (PEP) -** The PEP is usually implemented within a functional cluster and will query a PDP for allowance to perform an operation and will block the operation if necessary.
- **Policy Decision Point (PDP) -** The PDP should be implemented in a single locations (e.g. an OEM-specific application or an application provided by the stack vendor) and serve the PEP with authorization decisions. Furthermore, the PDP manages the authorization database(s).

7.2 Modeling

The application designer shall model each interaction point of an application with the ARA API. Therefore, every functional cluster shall define `PortInterfaces`representing its ARA API features as well as a set of sensible `GrantDesigns`. The `GrantDesigns` shall be structured such that security critical portions of the ARA API can be restricted. For example the ara::com API uses the `ServiceInterface` to represent its interaction points. The `ServiceInterface` itself consists of `Method`, `Event` and `Field` entities. Each of these entities may be subject to fine-grained access control restrictions. Hence, there are `GrantDesigns` for each of these entities.

Using a functional cluster's available meta model design elements, the application designer shall create a model consisting of:

- `PortPrototypes` referencing a functional cluster's `PortInterfaces` to express the need for using an ARA API
- `GrantDesigns` to request access to specific elements within a functional cluster's `PortInterface` or to the `PortInterface` itself

The integrator shall accept each requested access by creating an explicit `Grant` entity in the deployment model. The `Grant` shall reference the application designer's `GrantDesign`, a `Process`, and the functional cluster's respective deployment model entities (e.g. `SomeipServiceInterfaceDeployment`). If an integrator does not accept a requested access, a valid model cannot be created and therefore the integra-

tor and the application designer shall reconsider requesting the access or granting the access. This prevents using the IAM concept to achieve variant modeling.

Using the functional cluster's available meta model deployment elements, the integrator shall create a model consisting of:

- The application's runtime instances (`Process`)
- The functional cluster's deployment model
- The functional cluster's `Grants` linking the runtime instance to the protected asset

Given the deployment model the IAM-related entities can be transformed into a Processed Manifest for deployment to an ECU or for an integration into a software update package.

7.3 Runtime

There are several approaches to implementing the proposed Identity and Access Management concept. An exemplary approach will be outlined to provide a better understanding.

[SWS_IAM_01001]{DRAFT} **Separation of applications and Policy Enforcement Points** ⌈ The concept requires applications and PEPs to be separated by some operating system mechanism. ⌋*(RS_IAM_00002)*

The PEP may be part of the functional cluster's implementation or the operating system. One mechanism for separating applications from PEPs is hosting the code of the application and the functional cluster including the PEP in different processes. Given that example, the Adaptive Platform's ARA API implementation shall therefore implement an IPC from the application's process to the functional cluster's process.

[SWS_IAM_01002]{DRAFT} **Identification of an application** ⌈ An application's runtime instance shall be identifiable by the PEP. ⌋*(RS_IAM_00010)*

One mechanism for identifying a runtime instance of an application is to run the process as a distinct operating system user (e.g. Unix users). Given that example, Execution Management shall start application processes as distinct operating system users. Operating systems provide functionality to query peer credentials on an IPC channel (e.g. `getpeereid()` with Unix domain sockets). The Adaptive Platform implementation's IPC mechanism can encapsulate this functionality and transfer the information to the Policy Enforcement Point. The Policy Enforcement Point shall translate the actual runtime identity (i.e. the Unix user ID) to the model's representation (i.e. the Process). The Policy Decision Point shall use the reference to the model's runtime instance and the referencing Grants to allow or deny access.

A Not applicable requirements

All mentioned requirements are applicable.