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	18-03

Document Change History			
Date	**Release**	**Changed by**	**Description**
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 AUTOSAR Adaptive Identity and Access Management as part of the functional cluster Security Management 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

- The topic of providing identity information of Adaptive Applications to PEPs is still under discussion. Requirements and specification details regarding Application ID / Application Instance ID and providing application identity in general may be affected by this discussion and may change accordingly.
- No complete description of IAM API parameters since the content of the parameters is still under discussion.

4.2 Assumptions

The integrator can configure a secure 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

There are currently no dependencies to other functional clusters.

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_0000]	No description	[SWS_IAM_02001]
[RS_IAM_00002]	Enforcement of access control shall happen within `Adaptive Platform Foundation`	[SWS_IAM_02009] [SWS_IAM_02010]
[RS_IAM_00003]	Applications shall be prevented from taking control over the AUTOSAR PEP	[SWS_IAM_01000]
[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_01000] [SWS_IAM_02000] [SWS_IAM_02001]
[RS_IAM_00008]	Access control decisions must 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_02005] [SWS_IAM_02006] [SWS_IAM_02007] [SWS_IAM_02008]
[RS_IAM_00010]	Adaptive applications shall only be able to use AUTOSAR Resources when authorized	[SWS_IAM_02003]
[RS_IAM_000107]	No description	[SWS_IAM_02001]
[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]	For each AUTOSAR Resource an access control policy shall be specifiable	[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]	Unique Application Instance ID	[SWS_IAM_02013]

7 Functional specification

The AUTOSAR Adaptive Platform organizes the software of the AUTOSAR Adaptive Foundation as functional clusters. These clusters offer common functionality as services to the applications. The Identity and Access Management (IAM) for AUTOSAR Adaptive Platform is such a functional cluster and is part of “AUTOSAR Runtime for Adaptive Applications” - ARA. The functional cluster may consist of multiple modules. It is responsible for verifying identities and managing access control to resources.

The Identity and Access Management provides the infrastructure to perform access control for intra-ECU and inter-ECU operations. This infrastructure consists of platform components and optionally of OEM-specific applications.

This specification includes the syntax of the API to integrate an OEM-specific application, the relationship of the API to the model and describes the semantics and behavior. The specification does not pose constraints on the internal architecture and implementation.

7.1 Architectural concepts

The Identity and Access Management of AUTOSAR Adaptive can be logically divided into the following components:

- **Policy Enforcement Point (PEP)**
  The PEP is usually implemented in a functional cluster software component and will query a PDP for allowance to perform an operation and will block the operation if necessary.
- **Policy Decision Point (PDP)**
  The PDP may be implemented in several locations (e.g. an OEM-specific application, an application provided by the stack vendor) and manages access rights.

The related software components can be divided into the following components:

- Language binding
- Optional OEM-specific PDP application
- Identity and Access Management daemon

There are several types of interfaces available in the context of the Identity and Access Management:

- **Public Interface**
  Part of the ARA and specified in this document. This is the standardized ara::iam API.
- **Protected Interface**
  Used for interaction between functional clusters. This may be a custom API but it can also re-use the Public Interface. using the Public Interface is encouraged.
- **Private Interface**
  Used for interaction within the module. These interfaces are not described in the specification and are implementation-specific.

For the design of the ARA API the following constraints apply:

- Support the independence of application software components from a specific platform implementation
- Make the API as lean as possible, no specific use cases are supported which could also be layered on top of the API

Therefore the API of the Identity and Access Management follows a specific set of design decisions:

- It uses a pure virtual API to integrate different PDP application through a unified interface

7.2 Integration of Applications and Identity and Access Management

Any Application may implement a Policy Decision Point. This can be done by implementing the interface PolicyDecisionPoint::Check and registering the implementation using PolicyDecisionPointManagement::RegisterPolicyDecisionPoint. This Application may then also be responsible for exchanging access rights information with remote Adaptive Platform instances and locally managing access rights information.

Using a secure local channel the implemented Policy Enforcement Points can query registered Policy Decision Points.

[SWS_IAM_01000] Secure local channel ⌈ A secure local channel must be configured for integrating a Policy Decision Point with the implemented Policy Enforcement Points. The security of the local channel must be enforced by the operating system. It should enforce access control such that only authorized Applications can register Policy Decision Points. ⌋*(RS_IAM_00003, RS_IAM_00005)*

The implementation of Policy Enforcement Points is specific to the Adaptive Platform implementation. However the Policy Enforcement Points need to format their query according to [SWS_IAM_02011], [SWS_IAM_02012], [SWS_IAM_02013], [SWS_IAM_02014] and [SWS_IAM_02015].

8 API specification

8.1 C++ language binding

8.1.1 Type Definitions

Name:	PolicyDecision		
Type:	Scoped Enumeration of uint8_t		
Range:	kGranted kDenied	0 1	-- --

Syntax:	enum class PolicyDecision : uint8_t {
	kGranted = 0,
	kDenied = 1
	};		

Header file:	policy_decision.hpp		
Description:	Defines the decision made by a Policy Decision Point (see 7.1).		

Table 8.1: PolicyDecision

[SWS_IAM_02000] PolicyDecision Enumeration ⌈ Table 8.1 describes the enumeration PolicyDecision. ⌋*(RS_IAM_00005)*

8.1.2 Class Definitions

8.1.2.1 PolicyDecisionPoint

Service name:	PolicyDecisionPoint::Check	
Syntax:	`PolicyDecisionPoint Check( iam_string const& identity, iam_string const& domain, iam_string const& service, iam_string const& resource, iam_string const& operation );`	
Sync/Async:	Sync	
Parameters (in):	identity	Value of the requesting `Applications` identity.
	domain	Domain of the requesting `Application`.
	service	Service requested by the `Application`.
	resource	Resource requested by the `Application`.
	operation	Operation to be performed by the `Application`.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	`kGranted`	The request is granted.
	`kDenied`	The request is denied.
Exceptions:	None	
Description:	Interface for an `Policy Decision Point`.	

Table 8.2: PolicyDecisionPoint::Check

[SWS_IAM_02001] PolicyDecision::Check API ⌈ Table 8.2 describes the interface `PolicyDecisionPoint::Check`. ⌋(*RS_IAM_00005*, *RS_IAM_0000*, *RS_IAM_000107*)

[SWS_IAM_02002] iam_string type ⌈ The iam_string type is a type definition to allow varying implementation for stack vendors. It may be defined as:

```
using iam_string = std::basic_string<char>;
```

⌋()

[SWS_IAM_02003] Making a decision ⌈ When the `PolicyDecisionPoint::Check` is invoked the implementation shall check its internal rights database based on the inputs provided to decide whether the request is granted or not. If the request is granted, `kGranted` shall be returned. Otherwise `kDenied` shall be returned. ⌋(*RS_IAM_00010*, *RS_IAM_00011*)

[SWS_IAM_02004] Failing to make a decision ⌈ When the `PolicyDecisionPoint::Check` is invoked and the implementation cannot decide whether the request is granted or not, `kDenied` shall be returned. ⌋(*RS_IAM_00008*)

[SWS_IAM_02013] Identity value ⌈ The value of the parameter `identity` shall be a string representation of the `Application` identity the request is originating from. This could for example be the name of the `Adaptive Application`. ⌋(*RS_IAM_00011*, *RS_IAM_00012*, *RS_IAM_00014*, *RS_IAM_00015*)

[SWS_IAM_02012] Domain value ⌈ The value of the parameter `domain` shall be a string representation of the domain the request is originating from. This could for example be the name of the machine. ⌋*(RS_IAM_00011, RS_IAM_00012)*

[SWS_IAM_02011] Service value ⌈ The value of the parameter `service` shall be a concatenation of the protocol and the hexadecimal representation of the service identifier separated by a colon. An example for the SOME/IP service 0x4711 could be "someip:0x4711". For REST could be "rest:hostname" ⌋*(RS_IAM_00011, RS_IAM_00012)*

[SWS_IAM_02014] Resource value ⌈ The value of the parameter `resource` shall be the representation of the resource on which the operation is requested. For SOME/IP this could be the hexadecimal representation of the method ID. For REST this could be the URI of the resource the operation is intended to be performed on. ⌋ *(RS_IAM_00011, RS_IAM_00012)*

[SWS_IAM_02015] Operation value ⌈ The value of the parameter `operation` shall represent the operation that should be invoked. For SOME/IP this can be left empty since this is encoded in the method ID. For REST this could be any of the supported verbs e.g. "get". ⌋*(RS_IAM_00011, RS_IAM_00012)*

8.1.2.2 PolicyDecisionPointManagement

Service name:	PolicyDecisionPointManagement::PolicyDecisionPointManagement	
Syntax:	`PolicyDecisionPoint::PolicyDecisionPointManagement();`	
Sync/Async:	Sync	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Exceptions:	Implementation specific	In case the underlying IPC mechanism fails.
Description:	Creates an instance of PolicyDecisionPointManagement which opens the `Identity and Access Managements` communication channel (e.g. POSIX FIFO) for registering and offering policy decisions. Each `Application` shall create an instance of this class.	

Table 8.3: PolicyDecisionPointManagement::PolicyDecisionPointManagement

[SWS_IAM_02005] PolicyDecisionPointManagement::PolicyDecisionPointManagement API ⌈ Table 8.3 describes the interface `PolicyDecisionPointManagement::PolicyDecisionPointManagement`. ⌋*(RS_IAM_00011, RS_IAM_00009)*

Service name:	PolicyDecisionPointManagement::~PolicyDecisionPointManagement
Syntax:	`PolicyDecisionPoint::~PolicyDecisionPointManagement();`
Sync/Async:	Sync
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None

Exceptions:	None
Description:	Deletes the instance.

Table 8.4: PolicyDecisionPointManagement::~PolicyDecisionPointManagement

[SWS_IAM_02006] PolicyDecisionPointManagement::~PolicyDecisionPointManagement API ⌈ Table 8.4 describes the interface `PolicyDecisionPointManagement::~PolicyDecisionPointManagement`. ⌋*(RS_IAM_00011, RS_IAM_00009)*

Service name:	PolicyDecisionPointManagement::RegisterPolicyDecisionPoint	
Syntax:	`void RegisterPolicyDecisionPoint( PolicyDecisionPoint const& pdp );`	
Sync/Async:	Sync	
Parameters (in):	pdp	An implementation of a `Policy Decision Point`.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Exceptions:	Implementation specific	In case the underlying IPC mechanism fails.
Description:	Registers an actual implementation to perform policy decisions.	

Table 8.5: PolicyDecisionPointManagement::RegisterPolicyDecisionPoint

[SWS_IAM_02007] PolicyDecisionPointManagement::RegisterPolicyDecisionPoint API ⌈ Table 8.5 describes the interface `PolicyDecisionPointManagement::RegisterPolicyDecisionPoint`. ⌋*(RS_IAM_00011, RS_IAM_00009)*

Service name:	PolicyDecisionPointManagement::DeregisterPolicyDecisionPoint	
Syntax:	`void RegisterPolicyDecisionPoint( PolicyDecisionPoint const& pdp );`	
Sync/Async:	Sync	
Parameters (in):	pdp	An implementation of a `Policy Decision Point`.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Exceptions:	Implementation specific	In case the underlying IPC mechanism fails.
Description:	Deregisters an actual implementation.	

Table 8.6: PolicyDecisionPointManagement::DeregisterPolicyDecisionPoint

[SWS_IAM_02008] PolicyDecisionPointManagement::DeregisterPolicyDecisionPoint API ⌈ Table 8.6 describes the interface `PolicyDecisionPointManagement::DeregisterPolicyDecisionPoint`. ⌋*(RS_IAM_00011, RS_IAM_00009)*

[SWS_IAM_02009] Policy Decision Point and Enforcement Point communication ⌈ The local communication shall be facilitated by the `Application`'s PolicyDecisionPointManagement implementation. ⌋*(RS_IAM_00002)*

[SWS_IAM_02010] Policy Decision Point registration ⌈ By Registering a `Policy Decision Point` all queries will be routed to the registered `Policy Decision Point`'s `PolicyDecisionPoint::Check` method. A platform may support registering multiple `Policy Decision Points` or return an appropriate exception when attempting to register multiple `Policy Decision Points`. ⌋*(RS_IAM_00004, RS_IAM_00002)*

A Not applicable requirements

All mentioned requirements are applicable.