TTC技術レポート
Technical Report

TR-M2M-0016v2.0.0

認可アーキテクチャと
アクセス制御ポリシー

Authorization Architecture and
Access Control Policy

2016年11月30日制定

一般社団法人
情報通信技術委員会

THE TELECOMMUNICATION TECHNOLOGY COMMITTEE

TTC Telecommunication Technology Committee

TR-M2M-0016v2.0.0

Authorization Architecture and Access Control Policy

＜参考＞ [Remarks]

１．国際勧告等の関連 [Relationship with international recommendations and standards]

本技術レポートは、oneM2M で作成された Technical Report 0016 (Version 2.0.0) に準拠している。

[This Technical Report is transposed based on the Technical Report 0016 (Version 2.0.0) developed by oneM2M.]

２．作成専門委員会 [Working Group]

oneM2M 専門委員会 [oneM2M Working Group]

ONEM2M
TECHNICAL REPORT

Document Number	TR-0016-V-2.0.0
Document Name:	Study of Authorization Architecture for Supporting Heterogeneous Access Control Policies
Date:	2016-August-30
Abstract:	This a technical report that provides candidate security solutions for oneM2M authorization architecture, authorization procedures and access control policies.

This Specification is provided for future development work within oneM2M only. The Partners accept no liability for any use of this Specification.

The present document has not been subject to any approval process by the oneM2M Partners Type 1. Published oneM2M specifications and reports for implementation should be obtained via the oneM2M Partners' Publications Offices.

About oneM2M

The purpose and goal of oneM2M is to develop technical specifications which address the need for a common M2M Service Layer that can be readily embedded within various hardware and software, and relied upon to connect the myriad of devices in the field with M2M application servers worldwide.

More information about oneM2M may be found at: http//www.oneM2M.org

Copyright Notification

Notice of Disclaimer & Limitation of Liability

The information provided in this document is directed solely to professionals who have the appropriate degree of experience to understand and interpret its contents in accordance with generally accepted engineering or other professional standards and applicable regulations. No recommendation as to products or vendors is made or should be implied.

NO REPRESENTATION OR WARRANTY IS MADE THAT THE INFORMATION IS TECHNICALLY ACCURATE OR SUFFICIENT OR CONFORMS TO ANY STATUTE, GOVERNMENTAL RULE OR REGULATION, AND FURTHER, NO REPRESENTATION OR WARRANTY IS MADE OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR AGAINST INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS. NO oneM2M PARTNER TYPE 1 SHALL BE LIABLE, BEYOND THE AMOUNT OF ANY SUM RECEIVED IN PAYMENT BY THAT PARTNER FOR THIS DOCUMENT, WITH RESPECT TO ANY CLAIM, AND IN NO EVENT SHALL oneM2M BE LIABLE FOR LOST PROFITS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES. oneM2M EXPRESSLY ADVISES ANY AND ALL USE OF OR RELIANCE UPON THIS INFORMATION PROVIDED IN THIS DOCUMENT IS AT THE RISK OF THE USER.

Contents

This is a draft oneM2M document and should not be relied upon; the final version, if any, will be made available by oneM2M Partners Type 1.

1 Scope

The present document provides technical solutions for oneM2M authorization architecture, authorization procedures and access control policies. The present document also gives evaluations of these proposed technical solutions.

oneM2M TS-0003 [i.2] only defines a high level authorization architecture that describes its major components and general authorization procedure. The objective of the present document is to provide candidate security solutions related to authorization architecture, authorization procedures and access control policies.

The present document provides security solutions in the following three aspects:

- Detailed design of authorization architecture: This part investigates the interfaces among authorization components (e.g. procedures and parameters), how these components could be distributed in different oneM2M entities (i.e. different CSEs), and how to implement Role Based Access Control (RBAC) and token based access control.
- Supporting user specified access control policies: This part investigates how the oneM2M authorization system could be an extensible system that can support user-defined access control mechanisms and/or access control policy languages.
- Investigating existing access control policy languages: This part investigates if some standardized access control policy languages could become oneM2M recommended access control policy description languages.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the reference document (including any amendments) applies.

The following referenced documents are necessary for the application of the present document.

Not applicable.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the reference document (including any amendments) applies.

The following referenced documents are not necessary for the application of the present document but they assist the

user with regard to a particular subject area.

[i.1] oneM2M TS-0001: "Functional Architecture".

[i.2] oneM2M TS-0003: "Security Solutions".

[i.3] ANSI American national standard for information technology - role based access control. ANSI INCITS 359-2004, February 2004.

[i.4] NIST Special Publication 800-162 - Guide to Attribute Based Access Control (ABAC) Definition and Considerations.

[i.5] eXtensible Access Control Markup Language (XACML) Version 3.0. 22 January 2013. OASIS Standard.

[i.6] XACML v3.0 Core and Hierarchical Role Based Access Control (RBAC) Profile Version 1.0 Committee Specification 02 23 October 2014. OASIS Standard.

[i.7] Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0.

[i.8] oneM2M Drafting Rules.

NOTE: Available at

http://www.onem2m.org/images/files/oneM2M-Drafting-Rules.pdf.

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the definintions and abbreviations given in oneM2M TS-0011 apply:

4 Conventions,

The key words "Shall", "Shall not", "May", "Need not", "Should", "Should not" in the present document are to be interpreted as described in the oneM2M Drafting Rules [i.8].

5 Overview of authorization system

5.1 High level authorization architecture

Figure 5.1-1 provides a high level overview of a generic authorization architecture. This architecture comprises four subcomponents that are described as follows:

- Policy Enforcement Point (PEP):
  - PEP intercepts resource access requests, makes access control decision requests, and enforces access control decisions. The PEP coexists with the entity that needs authorization services.
- Policy Decision Point (PDP):
  - PDP interacts with the PRP and PIP to get applicable authorization polices and attributes needed to evaluate authorization policies respectively, and then evaluates access requests using authorization policies to render an access control decision. The PDP is located in the Authorization service.
- Policy Retrieval Point (PRP):
  - PRP obtains applicable authorization policies according to an access control decision request. These applicable policies should be combined in order to get a finial access control decision. The PRP is located in the Authorization service.
- Policy Information Point (PIP):
  - PIP provides attributes that are needed to evaluate authorization policies, for example the IP address of the requester, creation time of the resource, current time or location information of the requester. The PIP is located in the Authorization service.

The Authorization service may comprise any of the subcomponents: PDP, PRP and/or PIP. This means that the subcomponents PEP, PRP, PDP and PIP could be distributed across different nodes. For example the PEP is located in an ASN/MN and the PDP is located in the IN.

The present release 1 does not support separation of PRP and PIP on different CSE from PDP. The generic procedure described below is provided for information and to support further extensions, while clause 7 provides the details of authorization mechanisms in the current release.

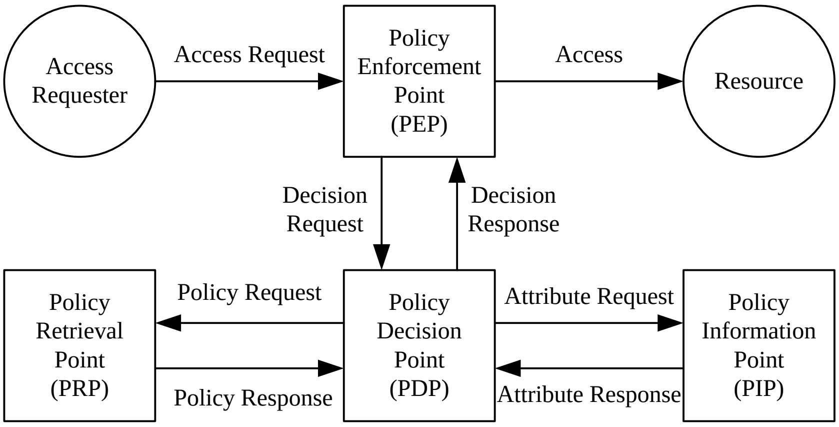

Figure 5.1-1: Overview of the authorization architecture

5.2 Generic authorization procedure

The generic authorization procedure is shown in figure 5.2-1.

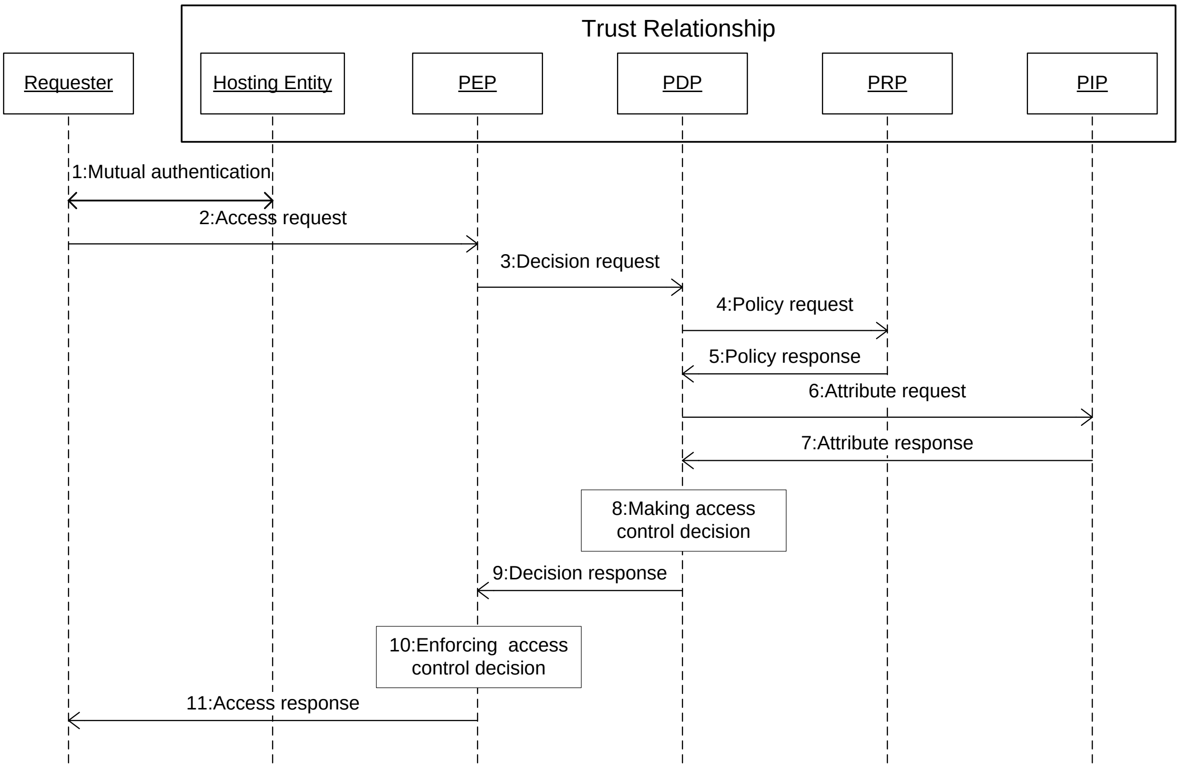

Figure 5.2-1: Authorization Procedure

Step 001: Mutual authentication (Pre-requisite).

Step 002: Access Requester sends an Access Request to the PEP.

Step 003: PEP makes an Access Control Decision Request according to the requester's Access Request, and sends the Access Control Decision Request to the PDP.

Step 004: PDP sends an Access Control Policy Request that is generated based on the Access Control Decision Request to the PRP.

Step 005: PRP finds all access control policies applicable to the access request and sends them back to the PDP. When multiple access control polices are involved, the PRP also provides a policy combination algorithm to combine multiple evaluation results into one finial result.

Step 006 PDP sends Attribute Request to the PIP, if any attributes are needed to evaluate these access control policies.

Step 007: PIP gets requested attributes and sends them back to the PDP.

Step 008: PDP evaluates Access Request using access control policies. When there are multiple applicable access control policies, the PEP needs to calculate a final Access Control Decision using the policy combination algorithm.

Step 009: PDP returns the Access Control Decision to the PEP.

Step 010: PEP enforces the access control decision, i.e. either forwards the Access Request to the resource or denies this access.

Step 011: PEP returns access result back to the Access Requester.

6 Detailed design of authorization architecture

6.1 Self-contained authorization

In a self-contained authorization system the PEP, PDP, PRP and PIP are all in the same CSE, and the messages exchanged between these authorization components are not crossing the oneM2M reference points Mca, Mcc and Mcn. So there is no specific standardization requirement about how to implement PEP, PDP, PRP and PIP, and the interactions between them.

6.2 Distributed authorization

6.2.1 Distributed authorization use cases

6.2.1.1 M2M gateway make access control decisions on behalf of m2m devices

Some constrained M2M Devices may be unable to evaluate the complex access control policy languages, such as those investigated in clause 8 "Investigating existing access control policy languages". These M2M Devices may be configured to request an M2M Gateway to assist with making access control decisions.

Here consider a scenario with two M2M Devices, Device 1 and Device 2, registered to a common M2M Gateway. Device 1 often interacts with M2M Devices that it has not encountered before, and so it frequently encounters situations where the Originator of the request cannot have been configured into the <accessControlPolicies> resources resident on Device 1. In this case, Device 1 has not encountered Device 2 previously, and so Device 1 requests the M2M Gateway to make an access control decision on behalf of Device 1. The relevant access control policies for Device 1 are not present on the M2M Gateway, so the M2M Gateway requests the relevant access control policies from M2M Server 1. When the M2M Gateway receives the access control policies, it realizes that it needs additional information about Device 2, so the M2M Gateway requests the relevant information from M2M Server 2. The M2M Gateway makes the access control decision and returns the decision result to Device 1. Figure 6.2.1.1-1 illustrates this process, which can be seen to map onto figure 5.1-1 "Overview of the authorization architecture".

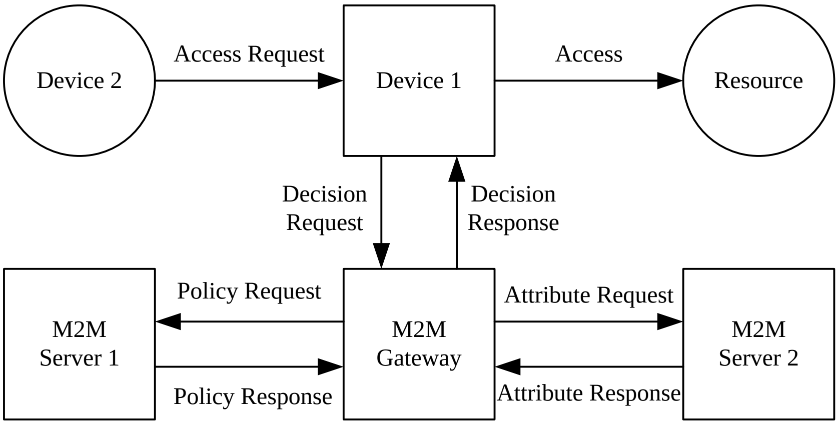

Figure 6.2.1.1-1: Use case scenario where an M2M Gateway makes authorization decisions on behalf of an M2M Device

Table 6.2.1.1-1 provides the mapping from the actors in the present use case scenario to functions in figure 5.1-1. Each authorization component (PEP, PDP, PRP and PIP) is on a distinct entity in this scenario. This motivates defining mechanisms for using oneM2M primitives enabling the following:

- a PEP entity requesting an access control decision from a distinct PDP entity;
- a PDP entity requesting relevant access control policies from a distinct PRP entity; and
- a PDP entity requesting relevant access control information from a distinct PIP entity.

Table 6.2.1.1-1: Mapping from actors in the use case scenario to the functions in Figure 5.1-1

Actor	Function in figure 5.1-1
Device 1	Policy Enforcement Point (PEP)
Device 2	Access Requestor
M2M Gateway	Policy Decision Point (PDP)
M2M Server 1	Policy Retrieval Point (PRP)
M2M Server 2	Policy Information Point (PIP)

6.2.2 Proposal 1: Using resource-based approach to implement distributed authorization

6.2.2.1 Introduction

According to the description in clause 8 of oneM2M TS-0001 [i.1], the general flow that governs the information exchange within a procedure is based on the use of Request and Response messages. The message applies to communications between an AE and a CSE which should cross the Mca reference point and among CSEs which should cross the Mcc reference point. Requests over the Mca and Mcc reference points, from an Originator to a Receiver should address the target resource or target attribute for the operation.

In the distributed authorization system the PEP, PDP, PRP and PIP might be located in different CSEs, so the communication between PEP, PDP, PRP and PIP should cross the Mcc reference point. The method of message exchange among these authorization components should conform to the oneM2M TS-0001 [i.1], i.e. the request message sent from one authoriztion component in one CSE to another authorization component in another CSE should address a resource.

According to the description in clause 9.2.2 of oneM2M TS-0001 [i.1], a virtual resource or a virtual attribute does not have a permanent representation in a CSE, they are used to trigger processing and/or retrieve results. So we can use virtual resources to exchange authorization messages among different CSEs, and at the same time, trigger a corresponding authorization process.

This clause describes a solution for distributed authorization using a newly defined <*authorization*> resource and its child resources over the Mcc and Mcc' reference points. The child resources of the <*authorization*> resource are <policyDecisionPoint>, <policyRetrievalPoint> and <policyInformationPoint>. These child resources are virtual resources that are used to trigger PDP process, PRP process and PIP process defined in oneM2M TS-0003 [i.2] respectively.

This clause also describes the management procedures for the <*authorization*> resource and its child resources.

6.2.2.2 Resources

6.2.2.2.1 Resource type *authorization*

The <*authorization*> resource represents the method for providing authorization related services. The <*authorization*> resource contains three child resources, they are <*policyDecisionPoint*>, <*policyRetrievalPoint*> and <*policyInformationPoint*>. These child resources are virtual resources that provide authorization functions of PDP, PRP and PIP defined in oneM2M TS-0003 [i.2] respectively. The <*authorization*> resource should be located directly under <*CSEBase*>.

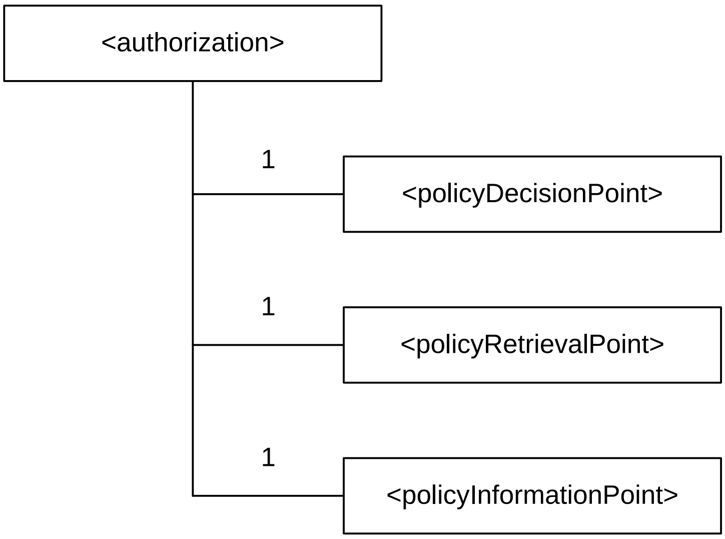

Figure 6.2.2.2.1-1: Structure of <*authorization*> resource

The <*authorization*> resource should contain the child resources specified in table 6.2.2.2.1-1.

Table 6.2.2.2.1-1: Child resources of <*authorization*> resource

Child Resources of <*authorization*>	Child Resource Type	Multiplicity	Description	<*authorizationAnnc*> Child Resource Types
[variable]	*<policyDecisionPoint>*	1	See clause 6.2.2.2.2	none
[variable]	*<policyRetrievalPoint>*	1	See clause 6.2.2.2.3	none
[variable]	*<policyInformationPoint>*	1	See clause 6.2.2.2.4	none

The <*authorization*> resource should contain the attributes specified in table 6.2.2.2.1-2.

Table 6.2.2.2.1-2: Attributes of <*authorization*> resource

Attributes of <*statsConfig*>	Multiplicity	RW/ RO/ WO	Description
resourceType	1	RO	See clause 9.6.1.3 of oneM2M TS-0001 [i.1] where this common attribute is described
resourceID	1	RO	See clause 9.6.1.3 of oneM2M TS-0001 [i.1] where this common attribute is described.
resourceName	1	WO	See clause 9.6.1.3 of oneM2M TS-0001 [i.1] where this common attribute is described.
parentID	1	RO	See clause 9.6.1.3 of oneM2M TS-0001 [i.1] where this common attribute is described.
authorizationPolicyIDs	1 (L)	RW	See clause 9.6.1.3 of oneM2M TS-0001 [i.1] where this common attribute is described
creationTime	1	RO	See clause 9.6.1.3 of oneM2M TS-0001 [i.1] where this common attribute is described
expirationTime	1	RW	See clause 9.6.1.3 of oneM2M TS-0001 [i.1] where this common attribute is described
lastModifiedTime	1	RO	See clause 9.6.1.3 of oneM2M TS-0001 [i.1] where this common attribute is described
labels	0..1 (L)	RW	See clause 9.6.1.3 of oneM2M TS-0001 [i.1] where this common attribute is described

6.2.2.2.2 Resource type *policyDecisionPoint*

The <*policyDecisionPoint*> resource is a virtual resource because it does not have a representation. It is the child resource of the <*authorization*> resource. When a RETRIEVE Request addresses the <*policyDecisionPoint*> resource, a PDP process is triggered. The access control decision request should be included in the Content parameter of the RETRIEVE Request, and the access control decision response should be included in the Content parameter of the RETRIEVE Response.

The <*policyDecisionPoint*> resource inherits access control policies that apply to the parent <*authorization*> resource.

6.2.2.2.3 Resource type *policyRetrievalPoint*

The <*policyRetrievalPoint*> resource is a virtual resource because it does not have a representation. It is the child resource of the <*authorization*> resource. When a RETRIEVE Request addresses the <*policyRetrievalPoint*> resource, a PRP process is triggered. The access control policy request should be included in the Content parameter of the RETRIEVE Request, and the access control policy response should be included in the Content parameter of the RETRIEVE Response.

The <*policyRetrievalPoint*> resource inherits access control policies that apply to the parent <*authorization*> resource.

6.2.2.2.4 Resource type *policyInformationPoint*

The <*policyInformationPoint*> resource is a virtual resource because it does not have a representation. It is the child resource of the <*authorization*> resource. When a RETRIEVE Request addresses the <*policyInformationPoint*> resource, a PIP process is triggered. The access control attribute request should be included in the Content parameter of the RETRIEVE Request, and the access control attribute response should be included in the Content parameter of the RETRIEVE Response.

The <*policyInformationPoint*> resource inherits access control policies that apply to the parent <*authorization*> resource.

6.2.2.3 Procedures

6.2.2.3.1 Introduction

This clause describes the management procedures for the <*authorization*> resource and its virtual child resources. These virtual child resources are <*policyDecisionPoint*>, <*policyRetrievalPoint*> and <*policyInformationPoint*> that are used to trigger a PDP process, a PRP process and a PIP process defined in oneM2M TS-0003 [2] respectively. Only Retrieve operation should be allowed on these virtual resources.

6.2.2.3.2 Create <*authorization*>

This procedure should be used to create a <*authorization*> resource.

Table 6.2.2.3.2-1: <authorization> CREATE

<authorization> CREATE	
Associated Reference Point	Mcc and Mcc'.
Information in Request message	All parameters defined in table 8.1.2-2 of oneM2M TS-0001 [i.1] apply with the specific details for: ***To:*** the address of the <*CSEBase*> where the <*authorization*> resource is intended to be Created. ***Content:*** attributes of the <*authorization*> resource as defined in clause 6.2.2.2.1-2.
Processing at Originator before sending Request	According to clause 10.1.1.1 of oneM2M TS-0001 [i.1].
Processing at Receiver	According to clause 10.1.1.1 of oneM2M TS-0001 [i.1] with the following additions: • Upon successful validation of the provided attributes, the Hosting CSE creates the <*authorization*> resource including its virtual child resources specified in table 6.2.2.2.1-1. • If there is a PDP process should be bound to the <*policyDecisionPoint*> virtual resource, then bind it to the <*policyDecisionPoint*> virtual resource, otherwise leave the binding void. The PDP process and the binding method are out of scope. • If there is a PRP process should be bound to the <*policyRetrievalPoint*> virtual resource, then bind it to the <*policyRetrievalPoint*> virtual resource, otherwise leave the binding void. The PRP process and the binding method are out of scope. • If there is a PIP process should be bound to the <*policyInformationPoint*> virtual resource, then bind it to the <*policyInformationPoint*> virtual resource, otherwise leave the binding void. The PIP process and the binding method are out of scope.
Information in Response message	According to clause 10.1.1.1 of oneM2M TS-0001 [i.1].
Processing at Originator after receiving Response	According to clause 10.1.1.1 of oneM2M TS-0001 [i.1].
Exceptions	According to clause 10.1.1.1 of oneM2M TS-0001 [i.1].

6.2.2.3.3 Retrieve <*authorization*>

This procedure should be used to retrieve the attributes of a <*authorization*> resource.

Table 6.2.2.3.3-1: <*authorization*> RETRIEVE

<authorization> RETRIEVE	
Associated Reference Points	Mcc and Mcc'
Information in Request message	According to clause 10.1.2 of oneM2M TS-0001 [i.1]
Processing at Originator before sending Request	According to clause 10.1.2 of oneM2M TS-0001 [i.1]
Processing at Receiver	According to clause 10.1.2 of oneM2M TS-0001 [i.1]
Information in Response message	According to clause 10.1.2 of oneM2M TS-0001 [i.1]
Processing at Originator after receiving Response	According to clause 10.1.2 of oneM2M TS-0001 [i.1]
Exceptions	According to clause 10.1.2 of oneM2M TS-0001 [i.1]

6.2.2.3.4 Update <*authorization*>

This procedure should be used to update the attributes of a <*authorization*> resource.

Table 6.2.2.3.4-1: <*authorization*> UPDATE

<authorization> UPDATE	
Associated Reference Points	Mcc and Mcc'
Information in Request message	According to clause 10.1.3 of oneM2M TS-0001 [i.1]
Processing at Originator before sending Request	According to clause 10.1.3 of oneM2M TS-0001 [i.1]
Processing at Receiver	According to clause 10.1.3 of oneM2M TS-0001 [i.1]
Information in Response message	According to clause 10.1.3 of oneM2M TS-0001 [i.1]
Processing at Originator after receiving Response	According to clause 10.1.3 of oneM2M TS-0001 [i.1]
Exceptions	According to clause 10.1.3 of oneM2M TS-0001 [i.1]

6.2.2.3.5 Delete <*authorization*>

This procedure should be used to delete a <*authorization*> resource.

Table 6.2.2.3.5-1: <*authorization*> DELETE

<authorization> DELETE	
Associated Reference Points	Mcc and Mcc'
Information in Request message	According to clause 10.1.4 of oneM2M TS-0001 [i.1]
Processing at Originator before sending Request	According to clause 10.1.4 of oneM2M TS-0001 [i.1]
Processing at Receiver	According to clause 10.1.4 of oneM2M TS-0001 [i.1]
Information in Response message	According to clause 10.1.4 of oneM2M TS-0001 [i.1]
Processing at Originator after receiving Response	According to clause 10.1.4 of oneM2M TS-0001 [i.1]
Exceptions	According to clause 10.1.4 of oneM2M TS-0001 [i.1]

6.2.2.3.6 Retrieve <*policyDecisionPoint*>

This procedure is used to trigger a PDP process that is bound to a <*policyDecisionPoint*> virtual resource.

Originator: The Originator should request to obtain an access control decision by using RETRIEVE operation on a <*policyDecisionPoint*> virtual resource from the Receiver. The Originator is a CSE.

Receiver: The Receiver should check if the Originator has RETRIEVE permission on the <*policyDecisionPoint*> virtual resource. Upon successful validation, the Receiver should make an access control decision based on access control policies. If there is no process bound to the <*policyDecisionPoint*> virtual resource, the Receiver should respond with an error.

Table 6.2.2.3.6-1: <*policyDecisionPoint*> RETRIEVE

<policyDecisionPoint> RETRIEVE	
Associated Reference Points	Mcc and Mcc'.
Information in Request message	According to clause 10.1.2 of oneM2M TS-0001 [i.1] with the following additions: ***To:*** The address of the <*policyDecisionPoint*> virtual resource. ***Content:*** The representation of the access control decision request.
Processing at Originator before sending Request	The Originator should create an access control decision request according to an access request. The Originator should request to obtain an access control decision by using the RETRIEVE operation on a <*policyDecisionPoint*> virtual resource which is the child resource of a <authorization> resource of the Receiver. The access control decision request should be included in the Content parameter of the Request message. The Originator should be a CSE.
Processing at Receiver	The Receiver should perform the following operations: • Check if the Originator has RETRIEVE permission on the <*policyDecisionPoint*> virtual resource. • Check the validity of the provided parameters. • Check if the <*policyDecisionPoint*> virtual resource is bound to a PDP process. • Upon successful validation, passing the access control decision request to the PDP process. • The PDP process contact a PRP to obtain applicable access control policies, and may also contact a PIP to obtain some access control attributes, and then make an access control decision. • The Receiver should create an access control decision response according to the access control decision and respond to the Originator. The access control decision response should be included in the Content parameter of the Response message.
Information in Response message	According to clause 10.1.2 of oneM2M TS-0001 [i.1] with the following additions: ***Content:*** The representation of the access control decision response.
Processing at Originator after receiving Response	The Originator should enforce the access control decision, i.e. either permit or deny the access to resource.
Exceptions	According to clause 10.1.2 of oneM2M TS-0001 [i.1] with the following: • There is no PDP process bound to the <*policyDecisionPoint*> virtual resource • The provided content of the access control decision request is not in line with the specified structure

6.2.2.3.7 Retrieve <*policyRetrievalPoint*>

This procedure is used to trigger a PRP process that is bound to a <*policyRetrievalPoint*> virtual resource.

Originator: The Originator should request to obtain access control policies by using RETRIEVE operation on a <*policyRetrievalPoint*> virtual resource from the Receiver. The Originator is a CSE.

Receiver: The Receiver should check if the Originator has RETRIEVE permission on the <*policyRetrievalPoint*> virtual resource. Upon successful validation, the Receiver should retrieve all applicable access control policies. If there is no process bound to the <*policyRetrievalPoint*> virtual resource, the Receiver should respond with an error.

Table 6.2.2.3.7-1: *<policyRetrievalPoint>* RETRIEVE

<policyRetrievalPoint> RETRIEVE	
Associated Reference Points	Mcc and Mcc'.
Information in Request message	According to clause 10.1.2 of oneM2M TS-0001 [i.1] with the following additions: ***To:*** The address of the <*policyRetrievalPoint*> virtual resource. ***Content:*** The representation of the access control policy request.
Processing at Originator before sending Request	The Originator should create an access control policy request according to an access control decision request. The Originator should request to obtain access control policies by using the RETRIEVE operation on a <*policyRetrievalPoint*> virtual resource which is the child resource of a <authorization> resource of the Receiver. The access control policy request should be included in the Content parameter of the Request message. The Originator should be a CSE.
Processing at Receiver	The Receiver should perform the following operations: • Check if the Originator has RETRIEVE permission on the <*policyRetrievalPoint*> virtual resource. • Check the validity of the provided parameters. • Check if the <*policyRetrievalPoint*> virtual resource is bound to a PRP process. • Upon successful validation, passing the access control policy request to the PRP process. • The PRP process retrieves all applicable access control policies according to the access control policy request. • The Receiver should create an access control policy response using the retrieved access control policies and respond to the Originator. The access control policy response should be included in the Content parameter of the Response message.
Information in Response message	According to clause 10.1.2 of oneM2M TS-0001 [i.1] with the following additions: ***Content:*** The representation of the access control policy response.
Processing at Originator after receiving Response	The Originator should evaluate the access control decision request using the retrieved the access control policies.
Exceptions	According to clause 10.1.2 of oneM2M TS-0001 [i.1] with the following: • There is no PRP process bound to the <*policyRetrievalPoint*> virtual resource. • The provided content of the access control policy request is not in line with the specified structure.

6.2.2.3.8 Retrieve <policyInformationPoint>

This procedure is used to trigger a PIP process that is bound to a <*policyInformationPoint*> virtual resource.

Originator: The Originator should request to obtain access control attributes by using RETRIEVE operation on a <*policyInformationPoint*> virtual resource from the Receiver. The Originator is a CSE.

Receiver: The Receiver should check if the Originator has RETRIEVE permission on the <*policyInformationPoint*> virtual resource. Upon successful validation, the Receiver should obtain the requested attributes. If there is no process bound to the <*policyInformationPoint*> virtual resource, the Receiver should respond with an error.

Table 6.2.2.3.8-1: <*policyInformationPoint*> RETRIEVE

<policyInformationPoint> RETRIEVE	
Associated Reference Points	Mcc and Mcc'.
Information in Request message	According to clause 10.1.2 of oneM2M TS-0001 [i.1] with the following additions: ***To:*** The address of the <*policyInformationPoint*> virtual resource. ***Content:*** The representation of the access control attribute request.
Processing at Originator before sending Request	The Originator should create an access control attribute request according to an access request. The Originator should request to obtain an access control attributes by using the RETRIEVE operation on a <*policyInformationPoint*> virtual resource which is the child resource of a <authorization> resource of the Receiver. The access control attribute request should be included in the Content parameter of the Request message. The Originator should be a CSE.
Processing at Receiver	The Receiver should perform the following operations: • Check if the Originator has RETRIEVE permission on the <*policyInformationPoint*> virtual resource. • Check the validity of the provided parameters. • Check if the <*policyInformationPoint*> virtual resource is bound to a PIP process. • Upon successful validation, passing the access control attribute request to the PIP process. • The PIP process obtains the requested access control attributes according to the access control attribute request. • The Receiver should create an access control attribute response using the obtained access control attributes and respond to the Originator. The access control attribute response should be included in the Content parameter of the Response message.
Information in Response message	According to clause 10.1.2 of oneM2M TS-0001 [i.1] with the following additions: ***Content:*** The representation of the access control attribute response.
Processing at Originator after receiving Response	The Originator should evaluate the access control decision request using the retrieved the access control policies and access control attributes.
Exceptions	According to clause 10.1.2 of oneM2M TS-0001 [i.1] with the following: • There is no PIP process bound to the <*policyInformationPoint*> virtual resource. • The provided content of the access control attribute request is not in line with the specified structure.

6.3 Message between authorization components

6.3.1 Proposal 1: Extending XACML and SAML for exchanging message between authorization components

6.3.1.1 Messages between PEP and PDP

6.3.1.1.1 Introduction of XACML <Request> element and <Response> element

XACML [i.5] defines a reference architecture that includes functions such as Policy Decision Points (PDPs), Policy Enforcement Points (PEPs), Policy Administration Points (PAPs), and Policy Information Points (PIPs) to control access. The functions of PEP and PDP defined in the oneM2M authorization architecture are the same as the functions of PEP and PDP defined in the XACML.

The format of an access decision request sent from a PEP to a PDP is specified by the <Request> element, and the format of an access decision response sent from a PDP to a PEP is specified by the <Response> element.

As XACML is primarily an Attribute Based Access Control system (ABAC), its <Request> element and <Response> element can express complicated access decision request and access decision response respectively. This proposal described in the following clauses suggests using XACML <Request> element and <Response> element to express the messages exchanged between PEP and PDP in the oneM2M authorization System.

6.3.1.1.2 Using XACML <Request> element

In XACML an access decision request is constructed by the <Request> element and its child elements: <Attributes>, <Attribute> and <Content>.

The <Request> element is defined as follows:

```
<xs:element name="Request" type="xacml:RequestType"/>
<xs:complexType name="RequestType">
    <xs:sequence>
        <xs:element ref="xacml:RequestDefaults" minOccurs="0"/>
        <xs:element ref="xacml:Attributes" maxOccurs="unbounded"/>
        <xs:element ref="xacml:MultiRequests" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="ReturnPolicyIdList" type="xs:boolean" use="required"/>
    <xs:attribute name="CombinedDecision" type="xs:boolean" use="required" />
</xs:complexType>
```

The <Attributes> elements are used to organize various request parameters into different attribute categories. Currently XACML defines four attribute categories:

- subject: this category is used to organize subject related attributes, such as Originator ID, roles and IP address of the Originator in the oneM2M System.
- resource: this category is used to organize resource related attributes, such as resource ID, URI and time of creation in the oneM2M System.
- action: this category is used to organize action related attributes, such as Create, Retrieve and Delete operation in the oneM2M System.
- environment: this category is used to organize environment related attributes, such as current date and current time.

The <Attributes> element is defined as follows:

```
<xs:element name="Attributes" type="xacml:AttributesType"/>
<xs:complexType name="AttributesType">
    <xs:sequence>
        <xs:element ref="xacml:Content" minOccurs="0"/>
        <xs:element ref="xacml:Attribute" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="Category" type="xs:anyURI" use="required"/>
    <xs:attribute ref="xml:id" use="optional"/>
</xs:complexType><xs:complexType name="SubjectType">
```

The <Attribute> elements are used to specify attributes in XACML. It contains attribute meta-data and one or more attribute values. The attribute meta-data comprises the attribute identifier and the attribute issuer. The <Attribute> element is defined as follows:

```
<xs:element name="Attribute" type="xacml:AttributeType"/>
<xs:complexType name="AttributeType">
    <xs:sequence>
        <xs:element ref="xacml:AttributeValue" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
    <xs:attribute name="Issuer" type="xs:string" use="optional"/>
    <xs:attribute name="IncludeInResult" type="xs:boolean" use="required"/>
</xs:complexType>
```

The <Content> element is used to hold additional sources of attributes in free form XML document format. The <Content> element is defined as follows:

```
<xs:element name="Content" type="xacml:ContentType"/>
<xs:complexType name="ContentType" mixed="true">
    <xs:sequence>
        <xs:any namespace="##any" processContents="lax"/>
    </xs:sequence>
</xs:complexType>
```

A proposal of parameter mapping from oneM2M access decision request to XACML access decision request is shown in the table 6.3.1.1.2-1.

Table 6.3.1.1.2-1: Parameters of oneM2M access request mapping to XACML access decision request

oneM2M request parameter	XACML Category	XACML Content/AttributeId	XACML DataType
to	resource	urn:oasis:names:tc:xacml:1.0:resource:resource-id	xs:anyURI
fr	subject	urn:oasis:names:tc:xacml:1.0:subject:subject-id	m2m:ID
role	subject	m2m:service-subcription-role	m2m:SRole-ID
op	action	urn:oasis:names:tc:xacml:1.0:action:action-id	m2m:operation
fc	resource	xacml:content	m2m:filterCriteria
rq_time	environment	urn:oasis:names:tc:xacml:1.0:subject:request-time	m2m:timestamp
rq_loc	subject	m2m:locationRegion	m2m:countryCode, m2m: circRegion
rq_ip	subject	urn:oasis:names:tc:xacml:3.0:subject:authn-locality:ip-address	m2m:ipv4, m2m:ipv6

The following is an example of using XACML decision request to express oneM2M authorization request which contains the information of Originator identity, role occupied by the Originator, resource URI, operation on the resource and the time of access.

```
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17 http://docs.oasis-
open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
targetNamespace="http://www.onem2m.org/xml/protocols"
xmlns:m2m="http://www.onem2m.org/xml/protocols"
ReturnPolicyIdList="false">
<Attributes Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
    <Attribute IncludeInResult="false" AttributeId="m2m:ID">
        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">//globalm2m.org/CSE1/123A38ZZY
        </AttributeValue>
    </Attribute>
        <Attribute IncludeInResult="false" AttributeId="m2m:service-subcription-role">
            <AttributeValue DataType="m2m:SRole-ID" >Software Management</AttributeValue>
            </Attribute>
        </Attributes>
    <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
        <Attribute IncludeInResult="false"
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id">
        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#anyURI">//cse1.mym2msp.org/</AttributeValue>
        </Attribute>
        </Attributes>
    <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
        <Attribute IncludeInResult="false" AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-
id" >
            <AttributeValue DataType="m2m:operation">Retrieve</AttributeValue>
            </Attribute>
        </Attributes>
    <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment">
    <Attribute IncludeInResult="false"
AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-date" >
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#date">2010-01-11</AttributeValue>
        </Attribute>
    </Attributes>
</Request>
```

6.3.1.1.3 Using XACML <Response> element

In XACML an access decision request is constructed by the <Response> element and its child element: <Result>.

The <Response> element is defined as follows:

```
<xs:element name="Response" type="xacml:ResponseType"/>
<xs:complexType name="ResponseType">
    <xs:sequence>
        <xs:element ref="xacml:Result" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
```

The <Result> element is defined as follows:

```
<xs:complexType name="ResultType">
    <xs:sequence>
        <xs:element ref="xacml:Decision"/>
        <xs:element ref="xacml:Status" minOccurs="0"/>
        <xs:element ref="xacml:Obligations" minOccurs="0"/>
        <xs:element ref="xacml:AssociatedAdvice" minOccurs="0"/>
        <xs:element ref="xacml:Attributes" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="xacml:PolicyIdentifierList" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
```

The meaning of these elements are:

- <Decision> element: describes the authorization decision that may be one of the following values:
  - "Permit": the requested access is permitted.
  - "Deny": the requested access is denied.
  - "Indeterminate": the PDP is unable to evaluate the requested access, for example missing attributes, syntax errors in the decision request, unable to retrieve policies, etc.
  - "NotApplicable: the PDP does not have any policy that is applicable to the decision request.
- <Status> element: indicates whether errors occurred during evaluation of the decision request and information about the errors.
- <Obligations> element: describes a list of obligations that should be fulfilled by the PEP.
- <AssociatedAdvice> element: describes a list of advice that might be used by the PEP.
- <Attributes> element: a list of attributes that were part of the request, which may be needed to be included in the decision response by the PEP.
- <PolicyIdentifierList>: the policy used by the PDP to evaluate the decision request may be returned if the PEP has such requirement.

The <Obligations> element can be used to pass an additional rule, policy or policy set that should be performed by the PEP. In the oneM2M System privacy related policies that should be performed by the PEP could be passed via the <Obligations> element.

The <AssociatedAdvice> element can be used to provide supplemental information about an access decision. In the oneM2M System <AssociatedAdvice> element could be used to pass various extension information to the PEP.

The following is an example of using XACML decision response to express oneM2M access response which expresses a permitted access decision.

```
<?xml version="1.0" encoding="UTF-8"?>
<Response xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd">
    <Result>
        <Decision>Permit</Decision>
        <Status>
            <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok" />
        </Status>
    </Result>
</Response>
```

6.3.1.2 Messages between PDP and PIP

6.3.1.2.1 Introduction of SAML

Security Assertion Markup Language (SAML) [i.7] is an XML-based, open-standard data format for exchanging authentication and authorization data between online business partners.

SAML uses assertions to exchange various security information. There are three assertions:

- Authentication assertion: it is used to exchange security information related to authentication.
- Attribute assertion: it is used for attribute query and attribute supplement.
- Authorization assertion: it is used for sending access decision request and response.

This proposal suggests using SAML attribute query to construct the attribute request messages sent from a PDP to a PIP, and using SAML attribute assertion to construct the attribute response messages sent from the PIP to the PDP.

6.3.1.2.2 Using SAML <AttributeQuery> element

This clause introduce how to use SAML <AttributeQuery> to construct the Attribute Request used in the oneM2M authorization system.

All SAML requests are derived from the abstract RequestAbstractType complex type. This type defines common attributes and elements that are associated with all SAML requests, through which request issuer, security information etc. could be added in. The RequestAbstractType complex type is defined as follows:

```
<complexType name="RequestAbstractType" abstract="true">
    <sequence>
        <element ref="saml:Issuer" minOccurs="0"/>
        <element ref="ds:Signature" minOccurs="0"/>
        <element ref="samlp:Extensions" minOccurs="0"/>
    </sequence>
    <attribute name="ID" type="ID" use="required"/>
    <attribute name="Version" type="string" use="required"/>
    <attribute name="IssueInstant" type="dateTime" use="required"/>
    <attribute name="Destination" type="anyURI" use="optional"/>
    <attribute name="Consent" type="anyURI" use="optional"/>
</complexType>
```

The <SubjectQuery> element is an extension point that allows new SAML queries to be defined that specify a single SAML subject. Its SubjectQueryAbstractType complex type adds the <saml:Subject> element to RequestAbstractType. the <SubjectQuery> element and its SubjectQueryAbstractType complex type are defined as follows:

```
<element name="SubjectQuery" type="samlp:SubjectQueryAbstractType"/>
<complexType name="SubjectQueryAbstractType" abstract="true">
    <complexContent>
        <extension base="samlp:RequestAbstractType">
            <sequence>
                <element ref="saml:Subject"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

In SAML the <AttributeQuery> element is used to request attributes about a given subject from an attribute provider. A successful response will be in the form of assertions containing attribute statements. This element is of type AttributeQueryType, which extends SubjectQueryAbstractType with the addition of the following element:

```
<element name="AttributeQuery" type="samlp:AttributeQueryType"/>
<complexType name="AttributeQueryType">
    <complexContent>
        <extension base="samlp:SubjectQueryAbstractType">
            <sequence>
                <element ref="saml:Attribute" minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

The <Attribute> element is used to describe the name and/or value of the requested attribute. The <Attribute> element is defined as follows:

```
<element name="Attribute" type="saml:AttributeType"/>
<complexType name="AttributeType">
    <sequence>
        <element ref="saml:AttributeValue" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="Name" type="string" use="required"/>
    <attribute name="NameFormat" type="anyURI" use="optional"/>
    <attribute name="FriendlyName" type="string" use="optional"/>
    <anyAttribute namespace="##other" processContents="lax"/>
</complexType>
```

The following is an example of using SAML AttributeQuery assertion to retrieve an AE's Service Subscription Roles. In this example the issuer of the attribute query is a PDP; the attribute authority is a PIP. The attribute requester is described in <Issuer> element, the attribute owner is described in <Subject> element, and requested attribute is described in <Attribute> element.

```
<samlp:AttributeQuery
    mlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    targetNamespace="http://www.onem2m.org/xml/protocols"
    xmlns:m2m="http://www.onem2m.org/xml/protocols"
    ID="hh89023-khl4580-wds2330"
    Version="2.0"
    IssueInstant="2015-09-10T13:25:30">
    <!--Attribute requester, for example a PDP-->
    <saml:Issuer Format="m2m:ID">
        //m2m.prov.com/CSE3219/C9886
    </saml:Issuer>
    <!--the subject whose attributes are required, for example an AE-->
    <saml:Subject>
        <saml:NameID Format="m2m:ID">
            //m2m.things.com/ab3f124a/Ca2efb3f4
        </saml:NameID>
    </saml:Subject>
    <!--the attribute is requested, for example the Service Subscription Role-->
    <saml:Attribute
        NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified"
        Name="m2m:SRole-ID"
        FriendlyName=" Service Subscription Role">
    </saml:Attribute>
</samlp:AttributeQuery>
```

6.3.1.2.3 Using SAML <Assertion> element

This clause introduces how to use SAML <Assertion> and <AttributeStatement> to construct the Attribute Response used in the oneM2M authorization system.

The <Assertion> element is of the AssertionType complex type. This type specifies the basic information that is common to all assertions. The RequestAbstractType complex type is defined as follows:

```
<element name="Assertion" type="saml:AssertionType"/>
<complexType name="AssertionType">
    <sequence>
        <element ref="saml:Issuer"/>
        <element ref="ds:Signature" minOccurs="0"/>
        <element ref="saml:Subject" minOccurs="0"/>
        <element ref="saml:Conditions" minOccurs="0"/>
        <element ref="saml:Advice" minOccurs="0"/>
        <choice minOccurs="0" maxOccurs="unbounded">
            <element ref="saml:Statement"/>
            <element ref="saml:AuthnStatement"/>
            <element ref="saml:AuthzDecisionStatement"/>
            <element ref="saml:AttributeStatement"/>
        </choice>
    </sequence>
    <attribute name="Version" type="string" use="required"/>
    <attribute name="ID" type="ID" use="required"/>
    <attribute name="IssueInstant" type="dateTime" use="required"/>
```

```
</complexType>
```

The <AttributeStatement> element describes a statement by the SAML authority asserting that the assertion subject is associated with the specified attributes. The AttributeStatementType complex type is defined as follows:

```
<element name="AttributeStatement" type="saml:AttributeStatementType"/>
<complexType name="AttributeStatementType">
    <complexContent>
        <extension base="saml:StatementAbstractType">
        <choice maxOccurs="unbounded">
            <element ref="saml:Attribute"/>
            <element ref="saml:EncryptedAttribute"/>
        </choice>
    </extension>
    </complexContent>
</complexType>
```

The following is an example of using SAML assertion and AttributeStatement to return an AE's Service Subscription Roles. In this example the issuer of the attribute query is a PDP; the attribute authority is a PIP. The attribute provider is described in <Issuer> element, and the value of the requested attribute is described in <AttributeStatement> element.

```
<saml:Assertion
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    targetNamespace="http://www.onem2m.org/xml/protocols"
    xmlns:m2m="http://www.onem2m.org/xml/protocols"
    ID="889967abd9847ghj986876k222"
    Version="2.0"
    IssueInstant="2015-09-07T14:25:23">
    <saml:Issuer>//www.m2mprovider.com/C3219</saml:Issuer>
    <saml:Subject>
        <saml:NameID Format="m2m:ID">
            //m2m.things.com/ab3f124a/Ca2efb3f4
        </saml:NameID>
    </saml:Subject>
    <saml:AttributeStatement>
        <saml:Attribute
            NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified"
            Name="m2m:SRole-ID"
            FriendlyName="Service Subscription Role">
            <saml:AttributeValue xsi:type="xs:string">01-001</saml:AttributeValue>
        </saml:Attribute>
    </saml:AttributeStatement>
</saml:Assertion>
```

6.4 Implementing Role Based Access Control

6.4.1 Introduction of Role Based Access Control

The essence of Role Based Access Control (RBAC) is that permissions are assigned to roles rather than to individual users. Roles are created for various job functions, and users are assigned to roles based on their qualifications and responsibilities. Users obtain the corresponding permissions through assigned appropriate roles. Users can be easily reassigned from one role to another without modifying the underlying access structure.

The RBAC reference model [i.3] defined by ANSI is shown in figure 6.4.1-1. This reference model comprises four model components-Core RBAC, Hierarchical RBAC, Static Separation of Duty Relations, and Dynamic Separation of Duty Relations. The characteristics of these models are:

- **Core RBAC** defines a minimum collection of RBAC elements, element sets, and relations in order to completely achieve a Role-Based Access Control system. Core RBAC includes sets of five basic data elements called users (USERS), roles (ROLES), objects (OBS), operations (OPS), and permissions (PRMS). The RBAC model as a whole is fundamentally defined in terms of individual users being assigned to roles and permissions being assigned to roles. As such, a role is a means for naming many-to-many relationships among individual users and permissions. In addition, the core RBAC model includes a set of sessions (SESSIONS) where each session is a mapping between a user and an activated subset of roles that are assigned to the user.

- **Hierarchical RBAC** component adds relations for supporting role hierarchies (RH). Role hierarchies define inheritance relations among roles, whereby senor roles acquire the permissions of their juniors and junior roles acquire users of their seniors.

- **Static Separation of Duty (SSD) Relations** are used to prevent users from obtaining conflicting roles in both the presence and absence of role hierarchies.

- **Dynamic Separation of Duty (DSD) Relations** are used to prevent users from activating conflict roles in a session.

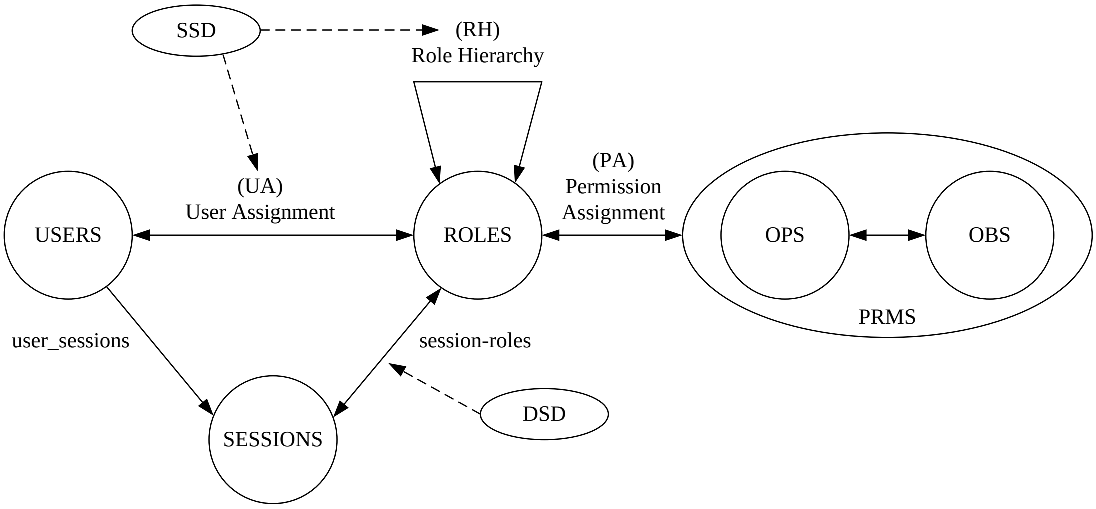

Figure 6.4.1-1: ANSI Role Base Access Control Model

6.4.2 General procedure of user-role assignment and role use

The general procedure of assigning role to an Originator and the originator uses the assigned role to access resource is shown in the figure 6.4.2-1 and described as follows:

1) An Originator sends a role token request to a token authority, which includes the role application information that describes what roles the Originator wants to apply.

2) The token authority checks the user-role assignment policies to determine if the requested roles can be assigned to the Originator. If it is permitted, the token authority issues an access token which contains the requested roles to the Originator.

3) The token authority sends the issued access token to the Originator.

4) The Originator sends a resource access request to a Hosting CSE, in which the issued access token is included.

5) The Hosting CSE (PEP in the Hosting CSE) generates an access decision request according to the Originator's resource access request, and then sends the request to a PDP, in which the access tokens of the Originator are included. The targeted PDP may be in the Hosting CSE or another CSE.

6) The PDP verifies the access tokens in the access decision request, extracts the roles from the valid access tokens, and then evaluates the applicable access control policies against the access decision request, including the roles for making an access control decision.

7) The PDP sends the access control decision via an access decision response to the Hosting CSE.

8) The Hosting CSE enforces the access control decision, i.e. either performs the resource access on behalf the Originator or denies the resource access.

9) The Hosting CSE returns the result of resource access back to the Originator.

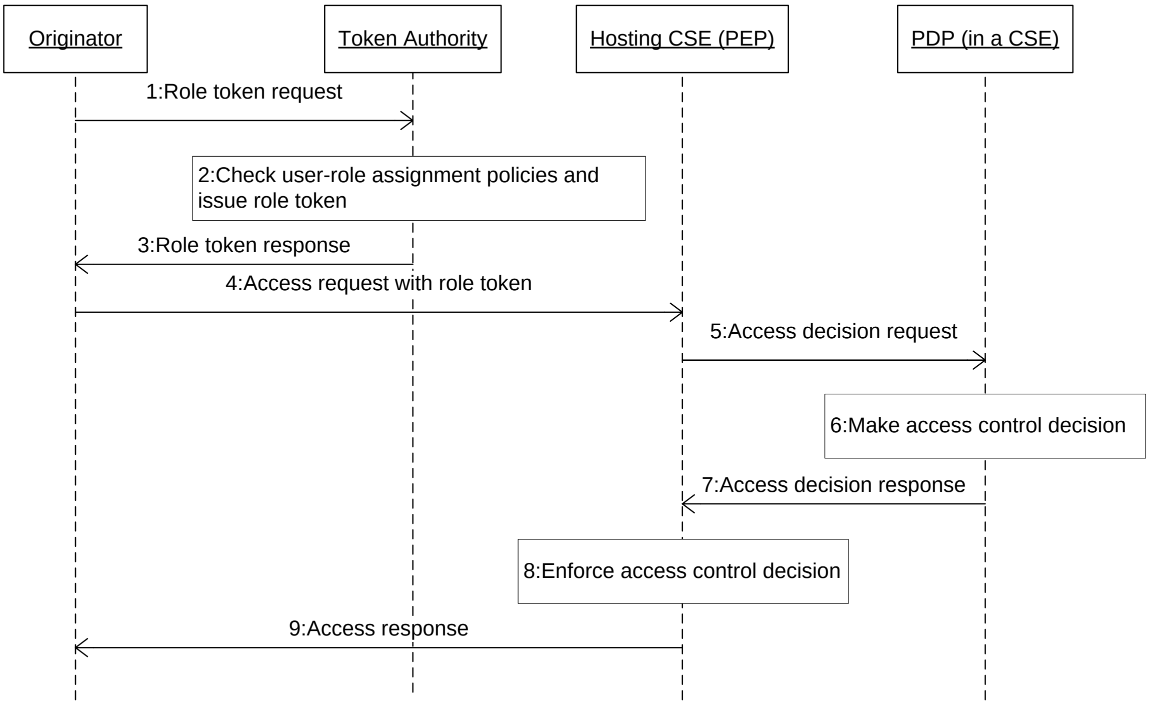
Originator
Token Authority
Hosting CSE (PEP)
PDP (in a CSE)
1:Role token request
2:Check user-role assignment policies and issue role token
3:Role token response
4:Access request with role token
5:Access decision request
6:Make access control decision
7:Access decision response
8:Enforce access control decision
9:Access response

Figure 6.4.2-1: General procedure of role issue and use

6.4.3 Solutions of implementing Role Based Access Control

6.4.3.1 Proposal 1: Solution of supporting Role Based Access Control

6.4.3.1.1 Role Based Access Control architecture

Figure 6.4.3.1.1-1 provides a high level overview of the Role Based Access Control architecture in the oneM2M System. This architecture comprises a new defined entity that is described as follows:

- Role Authority: It is responsible for assigning roles to an Originator according to role assignment policies. Role tokens that contain the assigned roles may also be issued by the Role Authority. Role token can provide integrity and/or confidentiality protection for its content. The Originator can be an AE or a CSE. The role assignment policies are however out of scope of the present document.

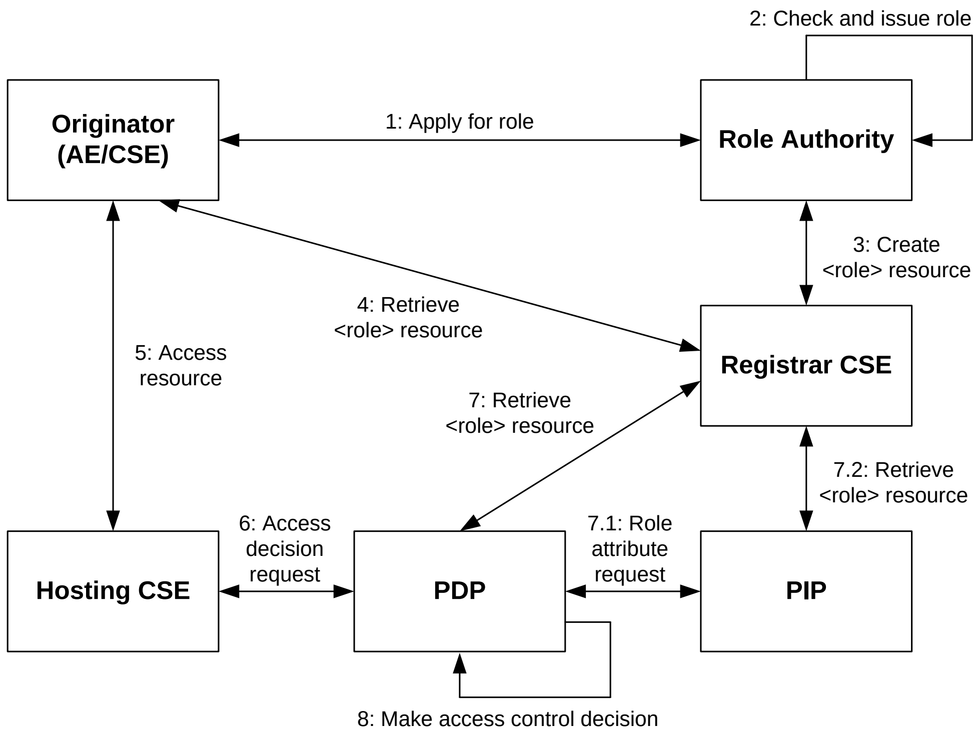
2: Check and issue role
Originator (AE/CSE)
1: Apply for role
Role Authority
3: Create <role> resource
4: Retrieve <role> resource
5: Access resource
Registrar CSE
7: Retrieve <role> resource
7.2: Retrieve <role> resource
6: Access decision request
7.1: Role attribute request
Hosting CSE
PDP
PIP
8: Make access control decision

Figure 6.4.3.1.1-1: Role Based Access Control architecture

The generic RBAC procedure is shown in figure 6.4.3.1.1-1 and described as follows:

Step 001: An Originator applies for a role from a Role Authority.

Step 002: The Role Authority checks if the role can be assigned to the Originator according to role assignment policies and issues the role to the Originator.

Step 003: The Role Authority create a <role> resource that is used to store the role information under the Originator's registration resource in the Originator's Registrar CSE.

Step 004: The Originator retrieves assigned roles from its registration resource.

Step 005: The Originator sends a resource access request in which roles are included to the Hosting CSE.

Step 006: The Hosting CSE sends an access decision request to a PDP.

Step 007: The PDP may need to retrieve role information of the Originator from the Originator's registration resource.

Step 008: The PDP verifies the Originator's role and makes an access control decision according to access control policies and roles.

6.4.3.1.2 Role token structure

The structure of role token is shown in figure 6.4.3.1.2-1, it contains the following data fields:

- version: version of the token format.
- tokenID: unique ID of the token.
- holder: ID of the token holder.
- issuer: ID of the token issuer.
- startTime: token valid from this time.

- expiryTime: token expired after this time.
- roleType: distinguish between different types of roles, e.g. M2M Service Roles or some other types of roles defined by M2M Application Service Providers.
- roleName: human readable name of the role.
- appCategories: List of M2M application identities that specify in which applications this role could be used for access control.
- role: ID of the role.
- extensions: Information defined and used by a specific application.

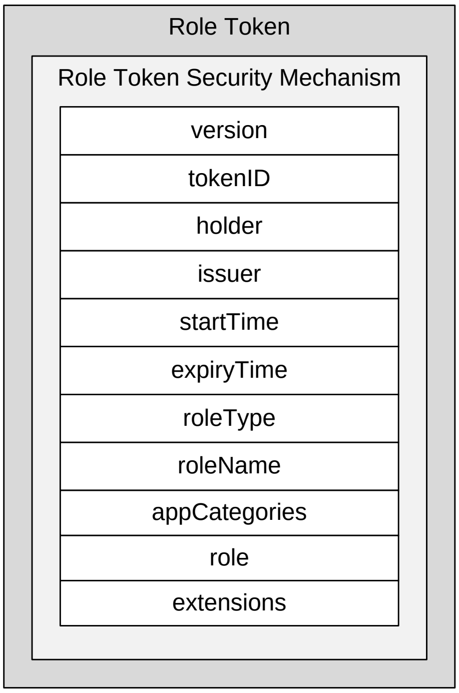

Figure 6.4.3.1.2-1: Structure of role token

6.4.3.1.3 Resource type *role*

The <*role*> resource represents a role assigned an AE or CSE. It is the child resource of an <*AE*> resource or a <*remoteCSE*> or directly under the <*CSEBase*> resource of the IN-CSE. One can get the information about what roles are assigned to an AE or a CSE through retrieving <*role*> resources under its <*AE*> resource or <*remoteCSE*> resource or the <*CSEBase*> resource of the IN-CSE.

An <*AE*> resource or a <*remoteCES*> or the <*CSEBase*> resource of the IN-CSE could have zero or multiple <*role*> child resources.

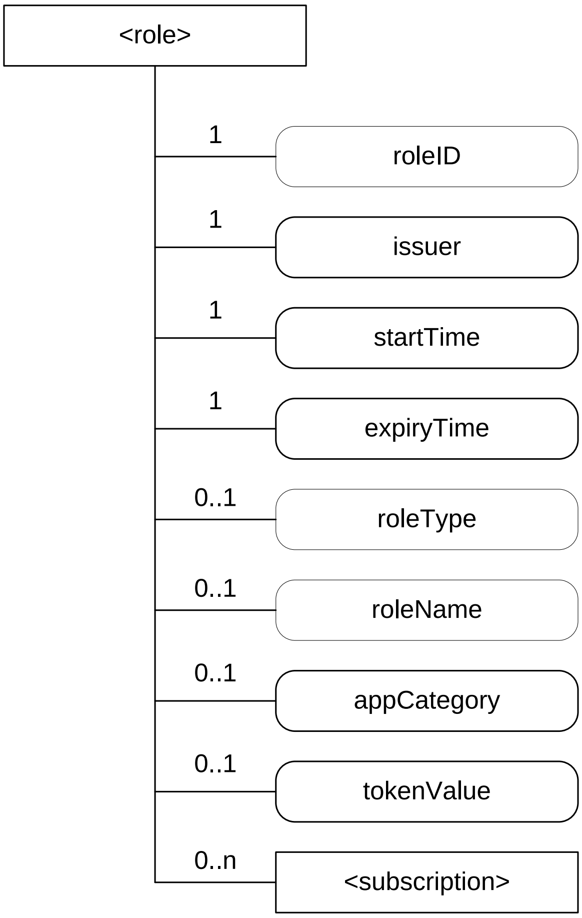

Figure 6.4.3.1.3-1: Structure of <AE> resource

The <*role*> resource should contain the child resources specified in table 6.4.3.1.3-1.

Table 6.4.3.1.3-1: Child resources of <*role*> resource

Child Resources of <*role*>	**Child Resource Type**	**Multiplicity**	**Description**	**<*role*> Child Resource Types**
[variable]	*<subscription>*	0..n	See clause 9.6.8 of oneM2M TS-0001 [i.1] where the type of this resource is described.	*[variable]*

The <*role*> resource should contain the attributes specified in table 6.4.3.1.3-2.

Table 6.4.3.1.3-2: Attributes of <*role*> resource

Attributes of <*role*>	**Multiplicity**	**RW/ RO/ WO**	**Description**
resourceType	1	RO	See clause 9.6.1.3 of oneM2M TS-0001 [i.1] where this common attribute is described.
resourceID	1	RO	See clause 9.6.1.3 of oneM2M TS-0001 [i.1] where this common attribute is described.
resourceName	1	WO	See clause 9.6.1.3 of oneM2M TS-0001 [i.1] where this common attribute is described.
parentID	1	RO	See clause 9.6.1.3 of oneM2M TS-0001 [i.1] where this common attribute is described.
expirationTime	1	RW	See clause 9.6.1.3 of oneM2M TS-0001 [i.1] where this common attribute is described.
accessControlPolicyIDs	0..1 (L)	RW	See clause 9.6.1.3 of oneM2M TS-0001 [i.1] where this common attribute is described. If no *accessControlPolicyIDs* is given at the time of creation, the *accessControlPolicyIDs* of the parent resource is linked to this attribute.
creationTime	1	RO	See clause 9.6.1.3 where this common attribute is described.
labels	0..1 (L)	RW	See clause 9.6.1.3 where this common attribute is described
lastModifiedTime	1	RO	See clause 9.6.1.3 where this common attribute is described.
roleID	1	RW	The identifier of the role.
issuer	1	RW	The identifier of the role issuer.
startTime	1	RW	Start time/date of the role can be used for access control. This attribute is mandatory for all role resources.
expiryTime	1	RW	End time/date of the role can be used for access control. This attribute is mandatory for all role resources.
roleType	0..1	RW	Distinguish between different types of roles, e.g. M2M Service Roles defined by M2M Service Providers, roles defined by M2M Application Service Providers in their specific M2M applications.
roleName	0..1	RW	Human readable name of the <*role*>.
appCategory	0..1	RW	Specify this role should be used for access control in which M2M application. The holder of the role could use this information to select applicable role for a resource access.
tokenValue	0..1	RW	Used to store a role token that contains the ID of a role that is assigned to the Originator. The role ID in the role token should be equal to the *roleID* attribute.

6.4.3.1.4 Role Based Access Control procedure without using role tokens

The general procedure of Role Based Access Control without using role tokens is shown in the figure 6.4.3.1.4-1 and described as follows:

Pre-configuration for the role issuance:

- The Originator has registered to the Registrar CSE.

Procedure of role token issuance and use:

1) The Originator sends a role application request to the Role Authority. This step may not be needed in some cases.

2) The Role Authority first checks if the applied role can be assigned to the Originator. If it is permitted, the Role Authority issues a role to the Originator.

3) The Role Authority sends a <role> resource creation request to the Originator's <AE>/<remoteCSE> resource which is in the Registrar CSE. The role assigned to the AE/CSE and the information pertaining to the role are included in the request.

4) The Registrar CSE checks the access control policies to determine if the Role Authority has the privilege of creating <role> resource in the target <AE>/<remoteCSE> resource. If it is permitted, the Registrar CSE creates the <role> resource.

5) The Registrar CSE returns the result of <role> resource creation back to the Role Authority.

6) The Role Authority returns the result of role issuance back to the Originator. The response may or may not contain the issued role and the information pertaining to this role.

7) The Originator sends a <role> resource retrieve request to its <AE>/<remoteCSE> resource in the Registrar CSE in order to get the assigned roles.

8) The Registrar CSE returns the retrieved <role> resources back to the Originator.

9) The Originator selects applicable roles according to the appCategory attribute of <role> resources, and includes them into the request sent to the Hosting CSE. The included role information should contain at least the role ID.

10) The PEP in the Hosting CSE generates an access decision request according to the request of the Originator, and sends the request to a PDP. The role information received from the Originator should be included in the request.

11) The PDP sends a role attribute request to a PIP in order to get the role information using the role ID, and the PIP further sends a <role> resource retrieve request to the Registrar CSE of the Originator. The PDP may directly send a <role> resource retrieve request to the Registrar CSE of the Originator instead of via a PIP.

12) The Registrar CSE retrieves the <role> resource in the target <AE>/<remoteCSE> resource according to the role ID, and returns the <role> resource back to the PIP or directly back to the PDP.

13) Based on the role information received from the Registrar CSE, the PDP checks if the role included in the access decision request is really assigned to the Originator and is still valid currently.

14) The PDP evaluates the resource access request of the Originator using access control policies and roles assigned to the Originator for making an access control decision.

15) The PDP returns the access control decision back to the PEP.

16) The Hosting CSE enforces the access control decision, i.e. either performs the resource access on behalf the Originator or denies the resource access.

17) The Hosting CSE returns the result of resource access back to the Originator.

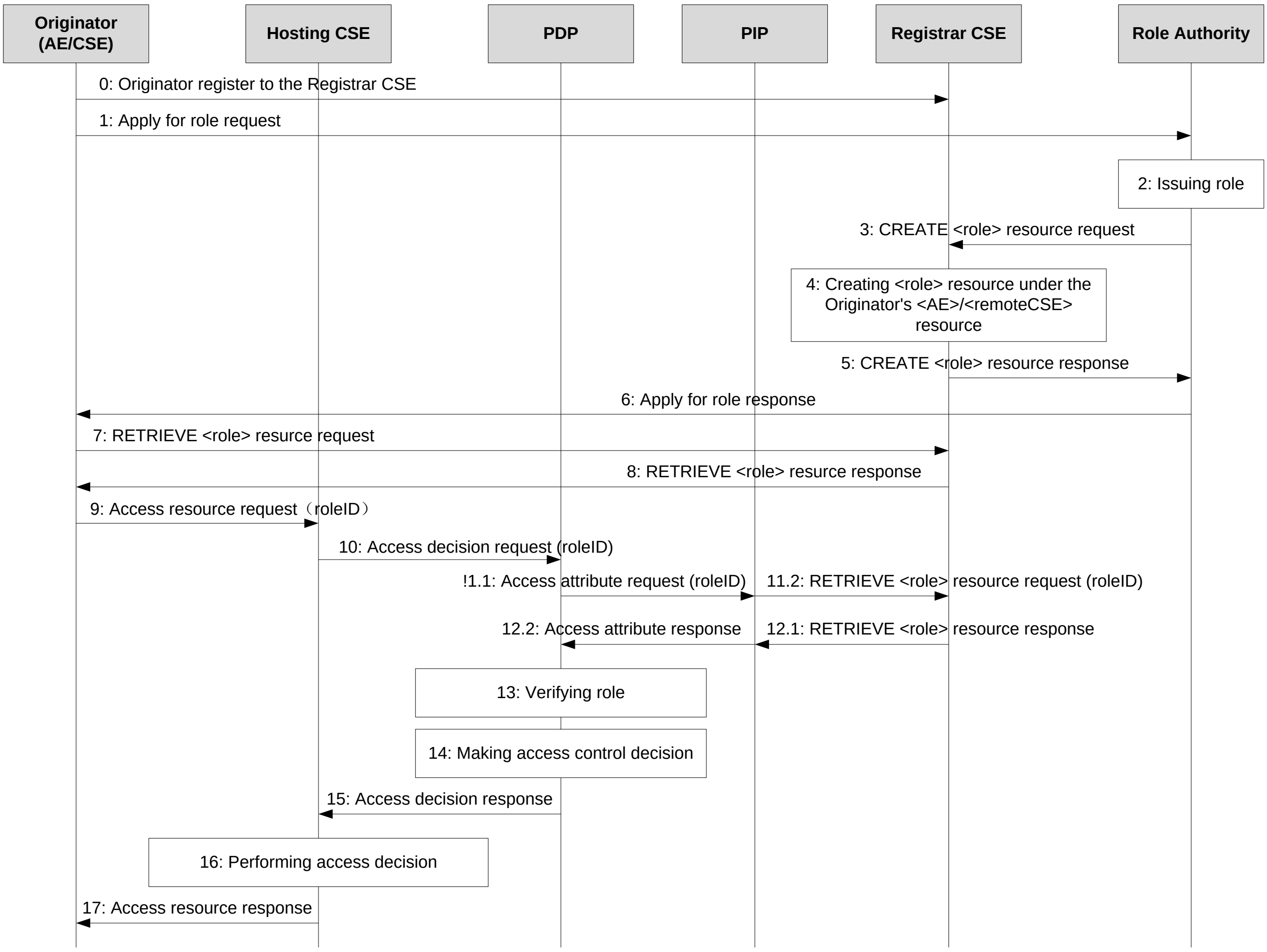
Originator (AE/CSE)
Hosting CSE
PDP
PIP
Registrar CSE
Role Authority
0: Originator register to the Registrar CSE
1: Apply for role request
2: Issuing role
3: CREATE <role> resource request
4: Creating <role> resource under the Originator's <AE>/<remoteCSE> resource
5: CREATE <role> resource response
6: Apply for role response
7: RETRIEVE <role> resurce request
8: RETRIEVE <role> resurce response
9: Access resource request（roleID）
10: Access decision request (roleID)
!1.1: Access attribute request (roleID)
11.2: RETRIEVE <role> resource request (roleID)
12.2: Access attribute response
12.1: RETRIEVE <role> resource response
13: Verifying role
14: Making access control decision
15: Access decision response
16: Performing access decision
17: Access resource response

Figure 6.4.3.1.4-1: Procedure of Role Based Access Control without using role tokens

6.4.3.1.5 Role Based Access Control procedure using role tokens

The general procedure of Role Based Access Control using role tokens is shown in the figure 6.4.3.1.5-1 and described as follows:

Pre-configuration for the token issuance:

- The Originator has registered to the Registrar CSE.
- The Role Authority has provided the security credentials used for verifying role tokens to the PDP.

Procedure of role token issuance and use:

1) The Originator sends a role application request to the Role Authority. This step may not be needed in some cases.

2) The Role Authority first checks if the applied role can be assigned to the Originator. If it is permitted, the Role Authority issues a role token that contains the applied role to the Originator.

3) The Role Authority sends a <role> resource creation request to the Originator's <AE>/<remoteCSE> resource which is in the Registrar CSE. The role token issued to the AE/CSE and information pertain to the token are included in the request.

4) The Registrar CSE checks the access control policies to determine if the Role Authority has the privilege of creating <role> resource in the target <AE>/<remoteCSE> resource. If it is permitted, the Registrar CSE creates the <role> resource.

5) The Registrar CSE returns the result of <role> resource creation back to the Role Authority.

6) The Role Authority returns the result of token issuance back to the Originator. The response may or may not contain the issued token and information pertaining to this role token.

7) The Originator sends a <role> resource retrieve request to its <AE>/<remoteCSE> resource in the Registrar CSE in order to get the assigned role tokens.

8) The Registrar CSE returns the retrieved <role> resources back to the Originator.

9) The Originator selects applicable roles according to the appCategory attribute of <role> resources, and includes them into the request sent to the Hosting CSE. The included role information should contain at least the role ID or role token.

10) The PEP in the Hosting CSE generates an access decision request according to the request of the Originator, and sends the request to a PDP. The role information received from the Originator should be included in the request.

11) In case only role ID is included in the access decision request, the PDP sends a role attribute request to a PIP in order to get the role information using the role ID, and the PIP further sends a <role> resource retrieve request to the Registrar CSE of the Originator. The PDP may directly send a <role> resource retrieve request to the Registrar CSE of the Originator instead of via a PIP. In case a role token is included in the access decision request, the PDP skips this step and the next step.

12) The Registrar CSE retrieves the <role> resource in the target <AE>/<remoteCSE> resource according to the role ID, and returns the <role> resource back to the PIP or directly back to the PDP.

13) The PDP verifies the received role token, the verification includes: the token is issued by a valid Role Authority and is still valid. If the token passes the verification, the PDP extracts the role from the token.

14) The PDP evaluates the resource access request of the Originator using access control policies and roles assigned to the Originator for making an access control decision.

15) The PDP returns the access control decision back to the PEP.

16) The Hosting CSE enforces the access control decision, i.e. either performs the resource access on behalf the Originator or denies the resource access.

17) The Hosting CSE returns the result of resource access back to the Originator.

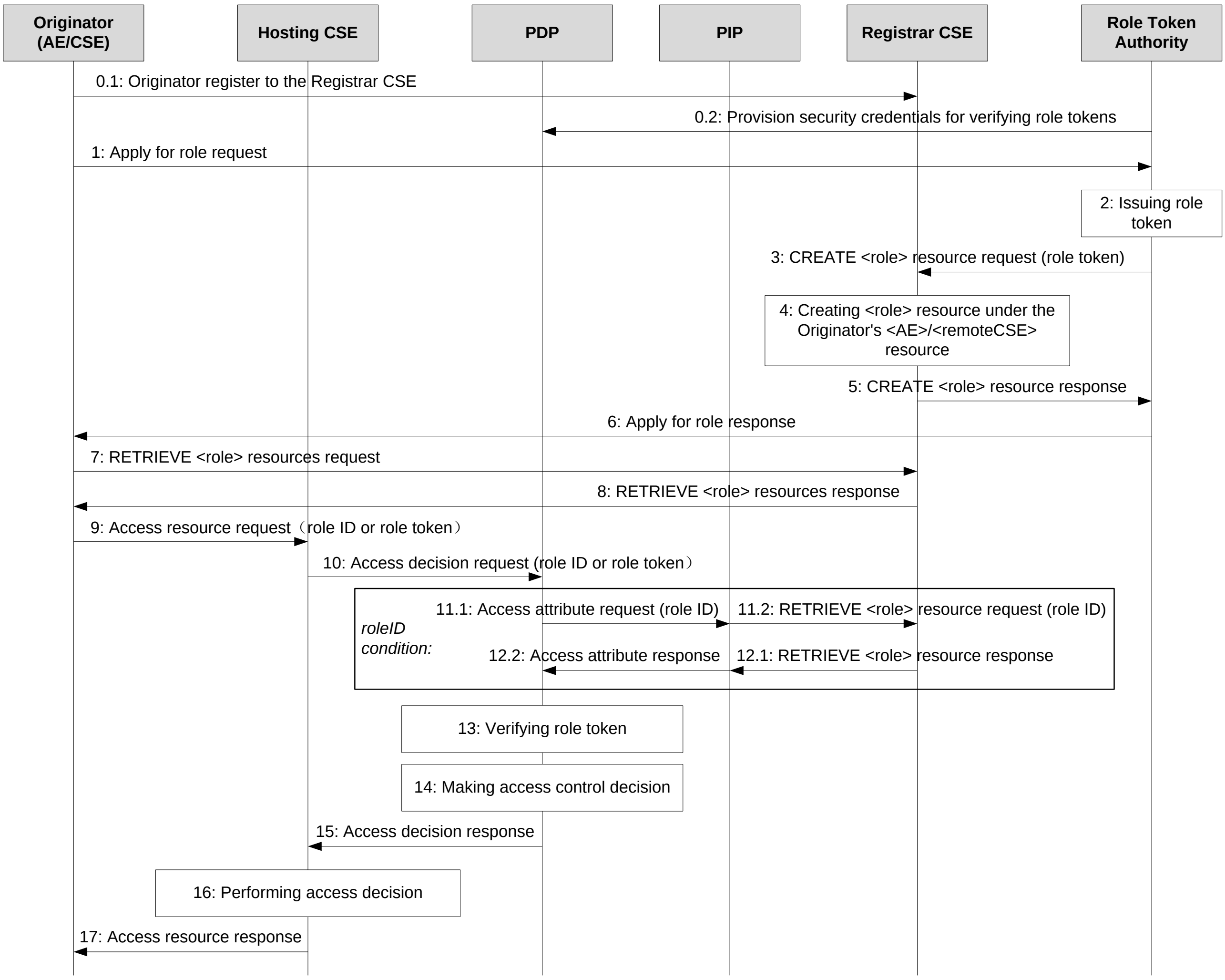
Originator (AE/CSE)
Hosting CSE
PDP
PIP
Registrar CSE
Role Token Authority
0.1: Originator register to the Registrar CSE
0.2: Provision security credentials for verifying role tokens
1: Apply for role request
2: Issuing role token
3: CREATE <role> resource request (role token)
4: Creating <role> resource under the Originator's <AE>/<remoteCSE> resource
5: CREATE <role> resource response
6: Apply for role response
7: RETRIEVE <role> resources request
8: RETRIEVE <role> resources response
9: Access resource request（role ID or role token）
10: Access decision request (role ID or role token）
roleID condition:
11.1: Access attribute request (role ID)
11.2: RETRIEVE <role> resource request (role ID)
12.1: RETRIEVE <role> resource response
12.2: Access attribute response
13: Verifying role token
14: Making access control decision
15: Access decision response
16: Performing access decision
17: Access resource response

Figure 6.4.3.1.5-1: Procedure of Role Based Access Control using role tokens

6.5 Implementing Attribute Based Access Control

6.5.1 Introduction of Attribute Based Access Control

The essence of Attribute Based Access Control (ABAC) is that accesses are granted to resources based on the attributes of requester, resources and environment conditions. In [i.4] the ABAC is defined as: an access control method where subject requests to perform operations on objects are granted or denied based on assigned attributes of the subject, assigned attributes of the object, environment conditions, and a set of policies that are specified in terms of those attributes and conditions.

According to [i.4], the core ABAC mechanism could be shown as figure 6.5.1-1. The major elements of ABAC are:

- **Attributes** are characteristics of the subject, object, or environment conditions. Attributes contain information given by a name-value pair.
- A **subject** is a human user or non-person entity, such as a device that issues access requests to perform operations on objects. Subjects are assigned one or more attributes.
- An **object** is a system resource for which access is managed by the ABAC system, such as devices, files, records, tables, processes, programs, networks, or domains containing or receiving information. It can be the resource or requested entity, as well as anything upon which an operation may be performed by a subject including data, applications, services, devices, and networks.

- An **operation** is the execution of a function at the request of a subject upon an object. Operations include read, write, edit, delete, copy, execute, and modify.

- An **access control policy** is the representation of rules or relationships that makes it possible to determine if a requested access should be allowed, given the values of the attributes of the subject, object, and possibly environment conditions.

- **Environment conditions**: operational or situational context in which access requests occur. Environment conditions are detectable environmental characteristics. Environment characteristics are independent of subject or object, and may include the current time, day of the week, location of a user, or the current threat level.

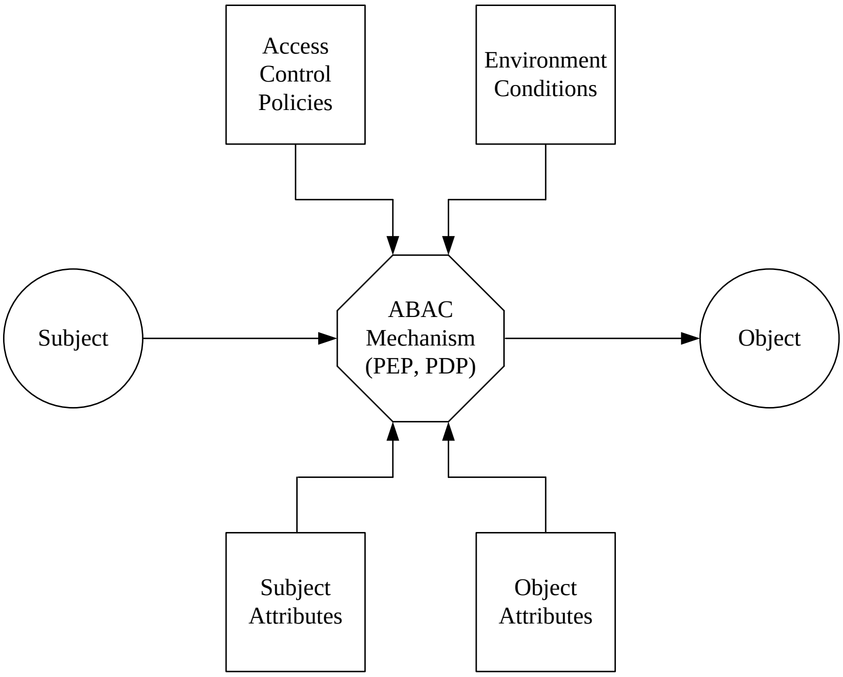

Figure 6.5.1-1: NIST Core ABAC Mechanism

Besides the ABAC data elements defined in [i.4], the ABAC model used in the oneM2M System should includes a set of sessions where each session is a mapping between a subject and an activated subset of attributes that are assigned to the subject. In the oneM2M System the subject attributes used by the ABAC mechanism for making an access control decision are those attributes which are activated in a session.

6.5.2 General procedure of Attribute Based Access Control

Attributes may be statically or dynamically assigned to the Originator:

- Statically assigned attributes: A PIP is configured with attributes that are statically assigned to the Originator.

- Dynamically assigned attributes: The Originator presents a token (with the request) which lists the attributes to be dynamically assigned to the Originator for that request.

The general procedure of ABAC in oneM2M System is shown in the figure 6.5.2-1 and described as follows:

1) (Steps 1 to 3 are applicable only if the Originator intends the PDP to consider dynamically assigned attributes). An Originator sends an access token request to a token authority describing the attributes the Originator wants to apply.

2) The token authority checks the access token issuing policies to determine if the requested attributes can be assigned to the Originator. If it is permitted, the token authority issues an access token which contains the requested attributes for the Originator.

3) The token authority sends the issued access token to the Originator.

4) The Originator sends a resource access request to a Hosting CSE, including (if applicable) the issued access tokens.

5) The Hosting CSE (PEP in the Hosting CSE) generates an access decision request according to the Originator's resource access request, and then sends the request to a PDP, including (if applicable) the access tokens. The targeted PDP may be in the Hosting CSE or another CSE.

6) The PDP retrieves the applicable access control policies, verifies the access tokens in the access decision request and extracts the attributes from the valid access tokens.

7) The PDP may send an attribute request to a PIP in order to obtain the attributes related to the Originator and/or the targeted resource.

8) The PIP returns the requested attributes to the PDP.

9) The PDP evaluates the applicable access control policies against the access decision request including the attributes of the Originator and resource to make an access control decision.

10) The PDP sends the access control decision via an access decision response to the Hosting CSE.

11) The Hosting CSE enforces the access control decision, i.e. either performs the resource access on behalf the Originator or denies the resource access.

12) The Hosting CSE returns the result of resource access back to the Originator.

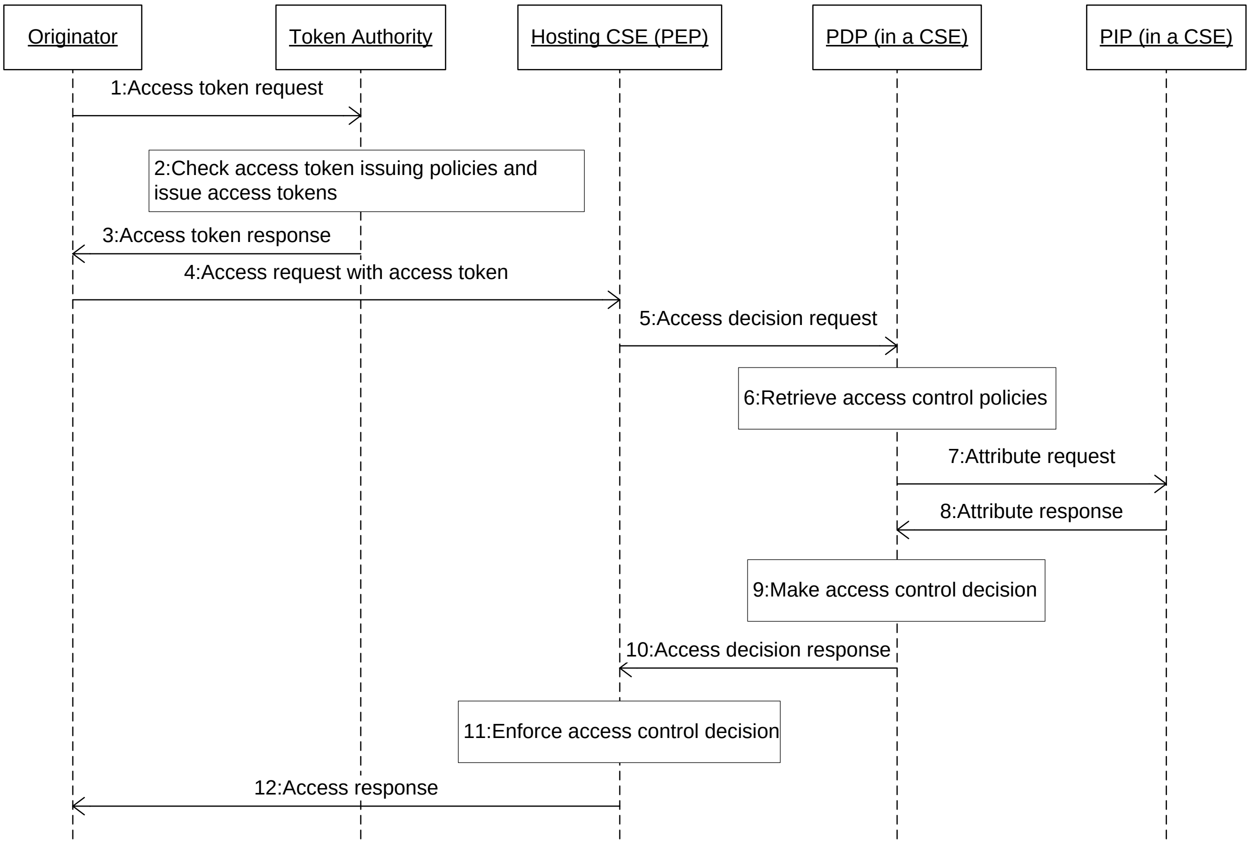
Originator
Token Authority
Hosting CSE (PEP)
PDP (in a CSE)
PIP (in a CSE)
1:Access token request
2:Check access token issuing policies and issue access tokens
3:Access token response
4:Access request with access token
5:Access decision request
6:Retrieve access control policies
7:Attribute request
8:Attribute response
9:Make access control decision
10:Access decision response
11:Enforce access control decision
12:Access response

Figure 6.5.2-1: General procedure of ABAC in oneM2M System

6.5.3 Solutions of implementing Attribute Based Access Control

There are two means for the oneM2M System to support ABAC policies:

- Using Extensible Access Control Markup Language (XACML) [i.5] that is consistent with ABAC.
- Using user specified access control policy language to express ABAC policies as described in clause 7.

7 Supporting user specified access control policies

7.1 Issues

According to the description in clause 9.6.2 of oneM2M TS-0001 [i.1] and in clause 7.1 of oneM2M TS-0003 [i.2], the current authorization solution does not support heterogeneous access control policies. As it is difficult to predicate all the authorization requirements of the oneM2M system and then design a versatile authorization system to satisfy all authorization requirements, it is reasonable for the oneM2M access control system to support user-defined access control mechanisms and/or access control policy languages.

7.2 Solutions

7.2.1 Proposal 1: Solution of supporting heterogeneous access control policies

7.2.1.1 Introduction

This clause describes a solution that extends the current oneM2M authorization system to support heterogeneous access control policies.

In the oneM2M authorization system there are two types of access control policies. One type is evaluated in PDP, another type is enforced in PEP such as privacy related access control policies. Access control policies enforced in PEP is also called obligation policy.

7.2.1.2 Redefined resource type *accessControlPolicy*

The redefined <*accessControlPolicy*> resource is shown in the figure 7.2.1.2-1.

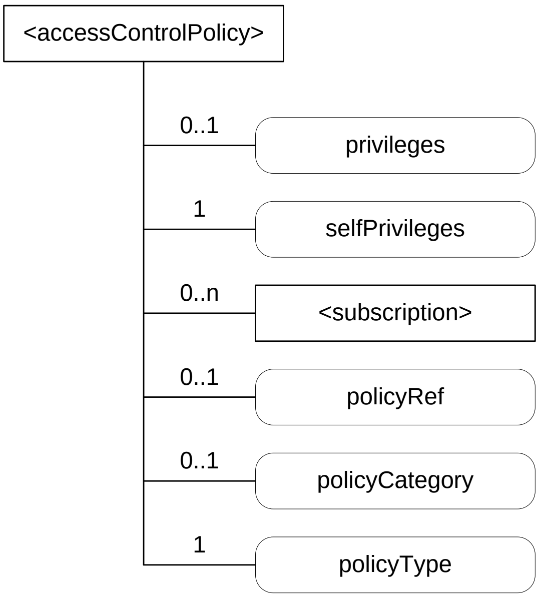

Figure 7.2.1.2-1: Redefined <*accessControlPolicy*> resource

In the redefined <*accessControlPolicy*> resource the modified and the new defined attributes are specified in table 7.2.1.2-1.

Table 7.2.1.2-2: New defined/modified attributes of <*accessControlPolicy*> resource

Attributes of <*accessControlPolicy*>	**Multiplicity**	**RW/ RO/ WO**	**Description**	**<*accessControlPolicyAnnc*> Attributes**
privileges	0..1	RW	A set of access control rules that applies to resources referencing this <*accessControlPolicy*> resource using the *accessControlPolicyID* attribute.	MA
policyRef	0..1	RW	A reference to an access control policy, from which access control policy can be retrieved.	MA
policyCategory	0..1	RW	PDP uses this attribute for selecting a suitable policy evaluator to evaluate the access control policy specified in *privileges* or *policyRef* attribute.	MA
policyType	1	RW	This attribute indicates the access control policy specified in *policyRef* attribute should be evaluated in a PDP or in a PEP.	MA

7.2.1.3 Generic procedure of evaluating heterogeneous access control policies

The generic procedure of evaluating heterogeneous access control policies is shown in figure 7.2.1.3-1 and described as follows:

- The process in PRP:
  - The PRP generates two policy sets named as ACPSet and ObligationSet respectively. The ACPSet is used for keeping the access control policies enforced by a PDP. The ObligationSet is used for keeping the obligation policies enforced by a PEP.
  - The PRP retrieves all applicable access control policies according to an access control policy request.
  - The PRP puts the applicable policies that should be enforced by the PDP into the ACPSet, and puts the applicable policies that should be enforced by the PEP into the ObligationSet.
  - The PRP returns both ACPSet and ObligationSet back to the PDP via an access control policy response.

- The process in PDP:
  - The PDP evaluates an access control decision request against the policies stored in the ACPSet, and makes an access control decision.
  - If the result of policy evaluation is "permit" and the ObligationSet is not empty, the PDP should return both the access control decision and the ObligationSet back to the PEP, otherwise only the access control decision should be returned back to the PEP via an access control decision response.
- The process in PEP:
  - The PEP enforces the access control decision, and fulfills the obligation policies in the ObligationSet if they accompany the access control decision.

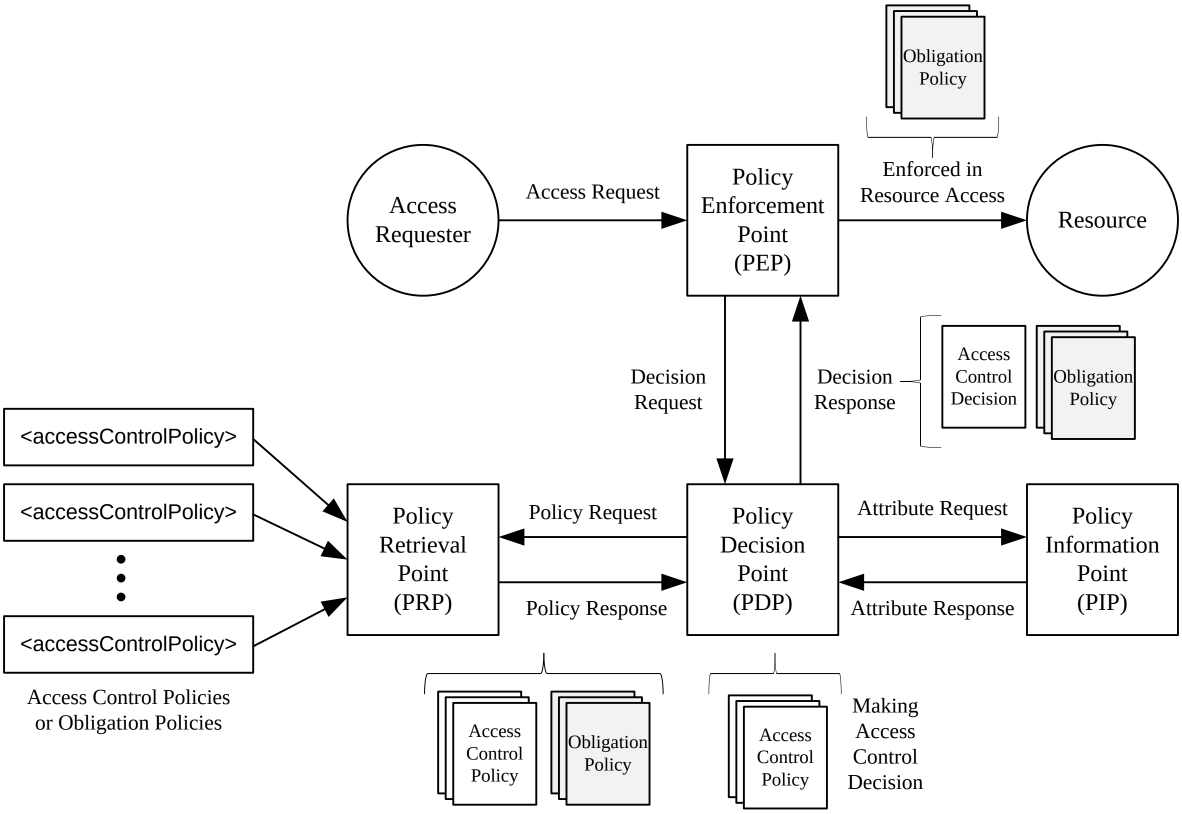

Figure 7.2.1.3-1: Generic procedure of evaluating heterogeneous access control policies

8 Investigating existing access control policy languages and proposals

8.1 Proposal 1: Using XACML

8.1.1 Introduction

eXtensible Access Control Markup Language (XACML) [i.5] is an XML-based access control language defined by the Organization for the Advancement of Structured Information Standards (OASIS). XACML access control framework conforms to the Attribute Based Access Control (ABAC).

8.1.2 Detailed descriptions

XACML policy structure is shown in figure 8.1.2-1, and described as follows.

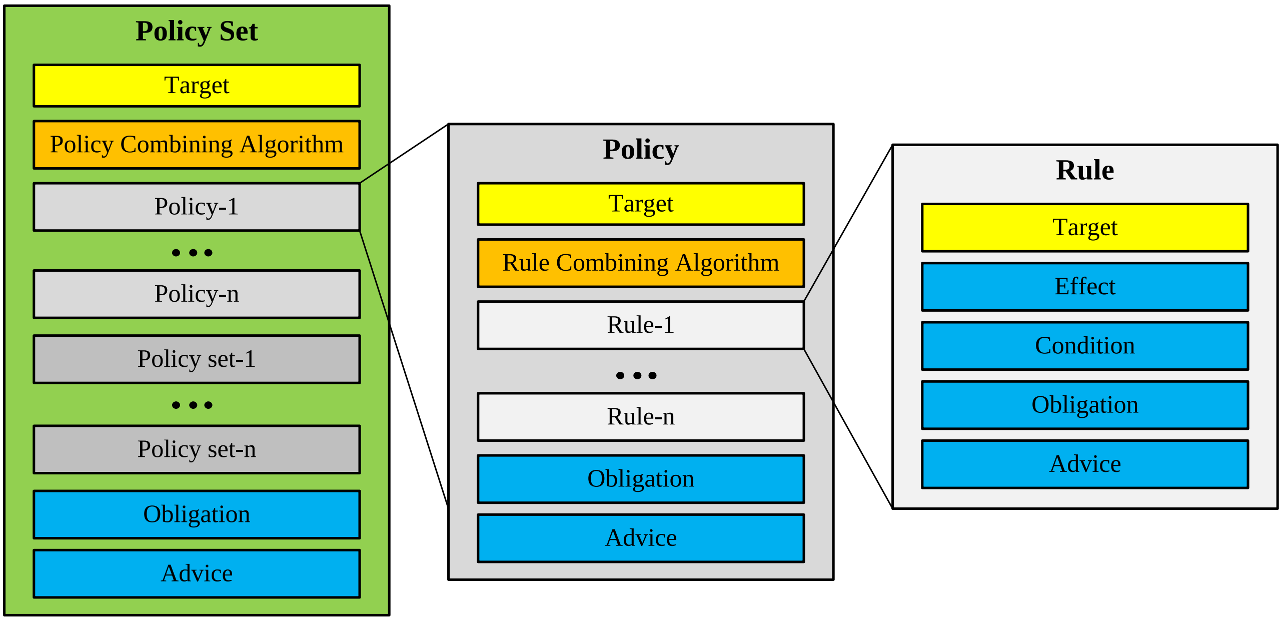

Figure 8.1.2-1: XACML policy structure

Structural elements

XACML is structured into 3 levels of elements:

- Policy set: It is the top level of a XACML policy. A policy set may contain the following components:
  - a target;
  - a policy-combining algorithm-identifier;
  - a set of policies;
  - a set of policy sets;
  - obligation expressions;
  - advice expressions.
- Policy: It is used to organize a set of rules. A policy may contain the following components:
  - a target;
  - a rule-combining algorithm-identifier;
  - a set of rules;
  - obligation expressions;
  - advice expressions.
- Rule: It is the most elementary unit of policy. A policy set may contain the following components:
  - a target;
  - an effect;
  - a condition;

- obligation expressions;
- advice expressions.

Targets

Policy set, policy and rule can all contain target elements. Target is basically a set of simplified conditions for the subject, resource, and action that should be met for a policy set, policy or rule to apply to a given request. Once a policy set, policy or rule is found to be applicable to a given request, the policies contained in the policy set, rules contained in the policy or the rule are evaluated to determine the access decision.

The structure of target is shown in figure 8.1.2-2. Each target may contain one to multiple matches, and each match contains one match function and two match values. One match value is preset in the rule, the other is obtained form a given request.

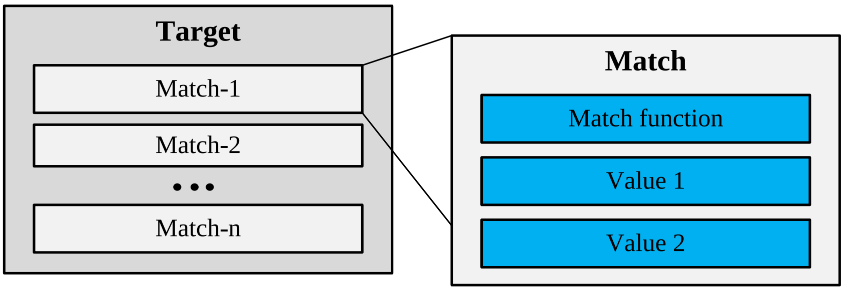

Figure 8.1.2-2: XACML target structure

Conditions

Conditions only exist in rules. It is used to check in which condition a rule can be used to evaluate a request.

The structure of condition is shown in figure 8.1.2-3. Each condition may contain one to multiple applies, and each apply contains one apply function and two match values. These match values could be preset in the rule or obtained dynamically, e.g. current time, IP address and so on.

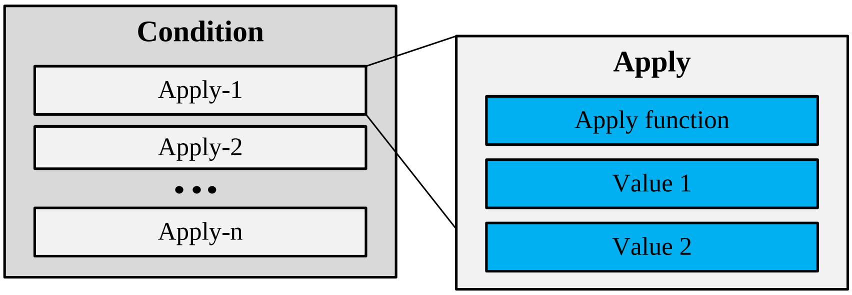

Figure 8.1.2-3: XACML condition structure

Attributes and categories

The values contained in the targets, match, apply and request are organized into four categories:

- Subject: It is the entity requesting access. A subject has one or more attributes.
- Resource: It is a data, service or system component. A resource has one or more attributes.
- Action: It defines the type of access requested on the resource. Actions have one or more attributes.
- Environment: It provides additional information, e.g. current time.

Effect

Effect is the intended consequence of the satisfied rule. It can either take the value Permit or Deny.

Obligations

An operation specified in a rule, policy or policy set which is provided to the PEP with the decision of the PDP. The PEP should perform it in conjunction with the enforcement of an authorization decision.

Advice

A supplementary piece of information in a rule, policy or policy set which is provided to the PEP with the decision of the PDP. In contrast to obligations, advice may be safely ignored by the PEP.

Combining algorithms

XACML defines two categories of combining algorithms that can be used to combine multiple rules or policies respectively. The rule combining algorithm defines a procedure for arriving at an access decision given the individual results of evaluation of a set of rules. Similarly, the policy combining algorithm defines a procedure for arriving at an access decision given the individual results of evaluation of a set of policies.

Request

A request can carry the following information:

- Subject attributes.
- Resource attributes.
- Action attributes.
- Environment attributes.

Response

A response can carry the following information:

- Decision.
- Status.
- Obligations.
- Advice.

8.1.3 Evaluation

From the perspective of the access control policy specification XACML has the following advantages and disadvantages.

Advantages:

- XACML is an access control policy language designed for all organizations.
- A standardized approach to authorization. Access control policies expressed in XACML are interoperable between different access control implementation by multiple vendors.
- The XACML model supports the separation of the access decision from the point of use. Authorization algorithms can be removed from the application logic of individual information systems.
- XACML conforms to the Attribute Based Access Control (ABAC).
- Role-based access control (RBAC) can be implemented in XACML.
- Support fine-grained access control.
- An access control system that supports XACML can be reused in other access control systems.
- XACML is extensible through defining new attribute identifiers, data types, policy combing algorithms, rule combing algorithms and functions of match or apply.

Disadvantages:

- XACML is verbose and complex in some ways.
- XACML does not standardize the interactions involving PAP and PIP.
- XACML does not standardize the policy administration and versioning.
- XACML does not support the specification of purpose or intent which is often associated with a privacy policy.
- XACML does not support dynamic authorization.
- There is some adoption, including large enterprises, but not broad, though commercial support is existent.

Recommendation:

For the reasons listed below, we should consider using XACML in the oneM2M system:

- XACML is a standardized access control policy language.
- There is no alternative to XACML for describing ABAC policy currently.

oneM2M Profile of XACML should be developed in order to achieve the interoperability in the oneM2M System.

8.2 Evaluation of oneM2M access control rule

8.2.1 Introduction

The descriptive ability of current oneM2M access control rule is evaluated in the following clauses.

8.2.2 Application scenario description

The oneM2M application scenario is shown in figure 8.2.2-1. CSE1 is a CSE. AE1 and AE2 are AE. Both AE1 and AE2 register to CSE1. CONT1 is a <*container*> resource in CSE1.

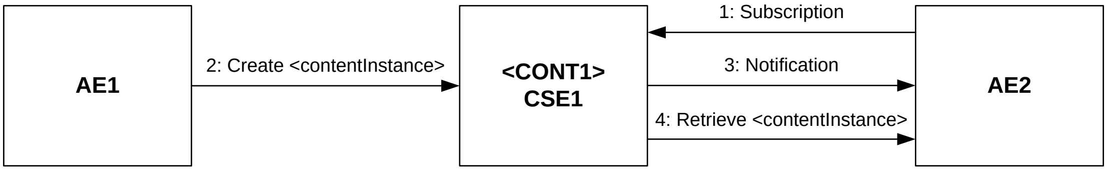

Figure 8.2.2-1: An oneM2M application scenario

In this oneM2M application AE1 and AE2 share data through CONT1 in CSE1. AE1 writes data into CONT1 through create <*contentInstance*> resources, and AE2 reads data from CONT1 through retrieve <*contentInstance*> resources from CONT1.

The basic procedure is:

1. AE2 makes a subscription to CONT1 in order to get notifications when new <*contenInstance*> resources are created in CONT1.
2. AE1 creates a <*contentInstance*> resource in CONT1.
3. CSE1 notifies AE2 after AE1 has created a <*contentInstance*> resource in CONT1.
4. AE2 retrieve the new created <*contentInstance*> resource from CONT1.

8.2.3 Access control rules and evaluation

The resource tree in CSE1 is shown in figure 8.2.3-1. ACP1 is a <*accessControlPolicy*> resource. ACP1 is associated to CONT1 through the *accessControlPolicyIDs* attribute of CONT1.

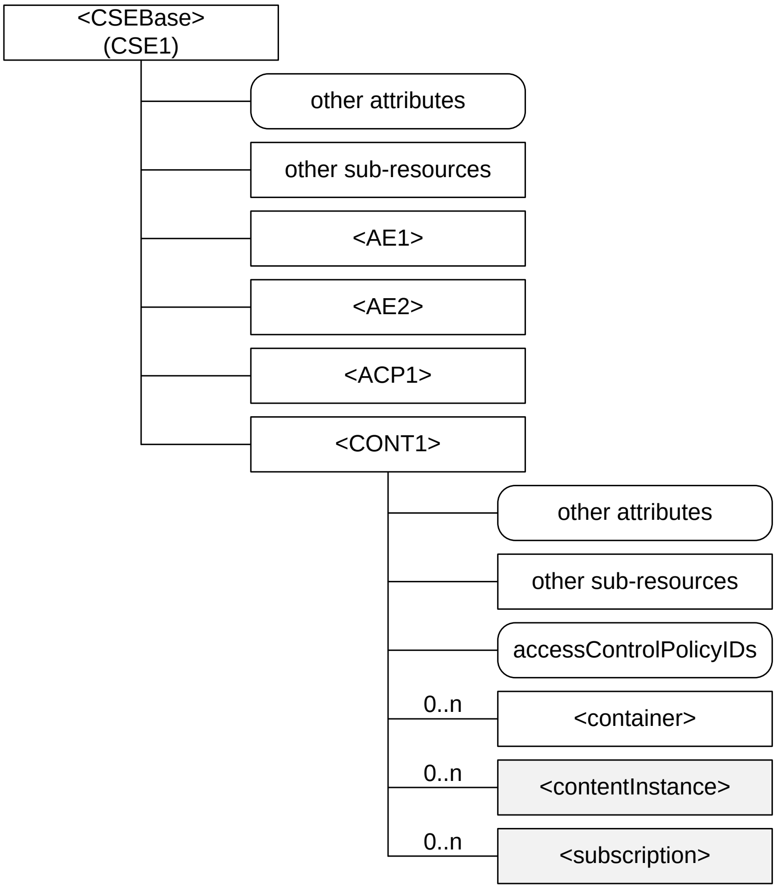

Figure 8.2.3-1: An example of resource tree

What we expect in access control are:

1. AE1 can create <*contentInstance*> resources in CONT1.
2. AE1 cannot create any other types of resources except <*contentInstance*> resources in CONT1.
3. AE2 can create <*subscriptio*n> resources for notifications about the change of <*contentInstance*> resources in CONT1.
4. AE2 cannot create any other types of resources except <*subscription*> resources in CONT1.
5. AE2 can retrieve <*contentInstance*> resources in CONT1.
6. AE2 cannot retrieve any other types of resources except <*contentInstance*> resources in CONT1.

In order to permit AE1 to create <*contentInstance*> resource and AE2 to create <*subscription*> resource in CONT1, the following rule should be applied.

Rule-1: [(AE1, AE2), (Create), ()]

However what we get is:

- AE1 and AE2 can create any child resources in CONT1. For example, both AE1 and AE2 can create <*container*> resources, and AE1 can create <*subscription*> resources. These are not what we expected.

In order to permit AE2 to retrieve <*contentInstance*> resources in CONT1, the following rule should be applied.

Rule-2: [(AE2), (Retrieve), ()]

However what we get is:

- AE2 can retrieve any attributes and/or child resources in CONT1. For example, AE2 can retrieve <*container*> resources if they do not have extra access control policies for their own, retrieve *accessControlPolicyIDs* attribute. These are not what we expected.

8.2.4 Conclusion

- More descriptive access control rules should be developed so that fine grained access control could be supported by the oneM2M System.

8.3 Proposal of new access control rule format

8.3.1 Introduction

This clause proposes a new access control rule format that can provide more expressive than the current access control rule format. The new rule format is an extension of the old one.

8.3.2 Rule format

8.3.2.1 Introduction

The set of access control rules represented in *privileges* and *selfPrivileges* attributes are comprised of 6-tuples (*accessControlResources*, *permittedAttributes*, *permittedChildResources*, *accessControlOriginators*, *accessControlContexts*, *accessControlOperations*) with parameters shown in table 8.3.2.1-1 which are further described in the following clauses.

If *privileges* attribute contains no 6-tuple then this represent an empty set of the access control rules.

The *selfPrivileges* attribute can contain at least one 6-tuples.

The CSE access granting mechanism could follow the procedure described in oneM2M TS-0003 [i.2] in clause 7.1 (Access Control Mechanism).

Table 8.3.2.1-1: Parameters in access-control-rule-tuple

Name	Description
accessControlResources	See clause 8.3.2.2
permittedAttributes	See clause 8.3.2.3
permittedChildResources	See clause 8.3.2.4
accessControlOriginators	See clause 9.6.2.1 of oneM2M TS-0001 [i.1]
accessControlContexts	See clause 9.6.2.2 of oneM2M TS-0001 [i.1]
accessControlOperations	See clause 9.6.2.3 of oneM2M TS-0001 [i.1]

8.3.2.2 *accessControlResources*

The *accessControlResources* is a mandatory parameter in an access-control-rule-tuple. It represents the set of resources that could be protected by this access control rule. The set of resources is described as a list of addresses of resources.

8.3.2.3 *permittedAttributes*

The *permittedAttributes* is an optional parameter in an access-control-rule-tuple that represents the set of attributes that are permitted to be accessed directly under the resources specified in the *accessControlResource* constraints by this access control rule. If this parameter does exist or it is empty, it means no attributes are allowed to be accessed directly under the resources within the *accessControlResources* constraints. Table 8.3.2.3-1 describes the supported types of parameters in *permittedAttributes*.

Table 8.3.2.3-1: Types of Parameters in *permittedAttributes*

Name	Description
attribute name	Resource attributes are allowed to be accessed directly under the resources specified within the *accessControlResources* constraints
All	Any resource attributes are allowed to be accessed directly under the resources specified within the *accessControlResources* constraints

8.3.2.4 *permittedChildResources*

The *permittedChildResources* is an optional parameter in an access-control-rule-tuple that represents the set of child resource types that are permitted to be accessed under the resources specified in the *accessControlResource* constraints by this access control rule. If this parameter does exist or it is empty, it means no child resource types are allowed to be accessed under the resources within the *accessControlResources* constraints. Table 8.3.2.4-1 describes the supported types of parameters in *permittedChildResources*.

Table 8.3.2.4-1: Types of Parameters in *permittedChildResources*

Name	Description
Resource Type	Resource types are allowed to be accessed under the resources specified within the *accessControlResources* constraints
All	Any resource types are allowed to be accessed under the resources specified within the *accessControlResources* constraints

8.3.3 Evaluation of the proposed oneM2M access control rule

The descriptive ability of the proposed access control rule is discussed in the following clauses using the application scenario described in the clause 8.2.

In order to permit AE1 to create <*contentInstance*> resources and not other resources in CONT1, the following rule could be applied.

New rule-1: [(CONT1); (); (contentInstance); (AE1); (Create); ()]

What we get is:

- AE1 can only create <*contentInstance*> resources in CONT1.

In order to permit AE2 to create <*subscription*> resources and not other resources in CONT1, the following rule could be applied.

New rule-2: [(CONT1); (); (subscription); (AE2); (Create); ()]

What we get is:

- AE2 can only create <*subscription*> resources in CONT1.

In order to permit AE2 to retrieve <*contentInstance*> resources in CONT1, the following rule could be applied.

New rule-3: [(CONT1); (); (contentInstance); (AE2); (Retrieve); ()]

What we get are:

- AE2 can only retrieve <*contentInstance*> resources in CONT1.

8.3.4 Conclusion

- Compared with current rule format the new rule format can more accurately describe access control rules, and can be used for supporting fine grained access control.

- The new access control rule contains the information of resources and Originators, so the access control rules belonging to different resources can be organized into one access control policy. This can simplify the access control policy management.

9 Privacy protection architecture using Privacy Policy Manager (PPM)

9.1 Introduction

The PPM is a personal data management framework based on the user's privacy preferences. The PPM creates access control policies from the user's privacy preference and protects the user's Personal Identifiable Information (PII) from service providers.

9.2 Relationship between components of PPM and oneM2M

The PPM has the following components. Detail of each component is explained in TR-0001-Use_Cases_Collection. This clause provides the relationship between the components and components of oneM2M.

1) Sophisticated consent mechanism for privacy policy:

    - When an end user subscribes to a service by an application service provider, the end user becomes a data subject, and the data subject creates a privacy preference and registers it on the PPM.

    - This function is for further study of oneM2M.

2) Functions of PDP or PRP:

    - PDP:

        - When an application service provider (ASP) accesses the data in the M2M platform, the PPM replies with an access decision that is decided from access control policies based on the user's privacy preference.

    - PRP:

        - When an ASP accesses the data in the M2M platform, PDP requests access control policies to the PPM. The PPM replies with the access control policies based on the user's privacy preference.

    - Components of PDP and PRP are defined in the following oneM2M documents:

        - oneM2M TS-0003 [i.2], clause 6.2.2 "Authorization Architecture".

        - oneM2M TS-0003 [i.2], clause 7.1 "Access Control Mechanism".

3) Traceability of personal data usage:

    - PPM stores the access log that records which ASPs access which kind of collected data.

    - This function is for further study of oneM2M, but this function can be implemented using components that are defined in oneM2M.

9.3 Privacy Policy Management in oneM2M architecture

9.3.0 Introduction

There are four procedures in the use of PPM. This clause explains relationships steps in the scenario and components of oneM2M.

- A data subject joins an M2M platform.
- A data subject subscribes to a service by an ASP.
- An ASP requests personal data that is stored in an M2M platform.
- The data subject checks the access log of his/her own personal data and requests the deletion to the ASP.

9.3.1 Actor

- Data subject:
    - An end user can make use of services on an M2M platform by subscribing to a service of an ASP which controls access to the M2M platform.
    - When an end user subscribes to a service by an ASP, the end user become a data subject.
- M2M Device (AE):
    - An M2M device collects various kinds of data, such as sensor.
    - An M2M device sends the data to an M2M gateway.
- M2M Gateway (MN-CSE):
    - Policy Enforcement Point (PEP):
        - PEP is one of the functions in the CSE.
    - Policy Decision Point (PDP):
        - PDP is one of the functions in the CSE.
    - Personal Data:
        - Personal data is information that can be used on its own or with other information to identify.
        - An M2M gateway collects and stores personal data from M2M devices.
        - Examples of personal data: Sensor data, Electrical power consumption, Operating state of air conditioner, etc.
- Application Service Provider (ASPAE):
    - An ASP provides services to an end user who joins the M2M platform.
    - An ASP requests personal data from an M2M platform in order to provide services.
- M2M Platform (IN-CSE):
    - Portal:
        - A portal is a kind of Web site or Web application in an M2M platform.
        - An end user accesses a portal to join an M2M platform. A data subject access the portal to subscribe to a service by an ASP.

- PPM:
  - The PPM stores access control policies based on user's privacy preference.
  - The PPM has functions of PDP or PRP.
  - Portal:
    - A portal is a kind of Web site or Web application in a PPM.
    - An end user accesses a portal to configure the end user's privacy preference.

9.3.2 Management flow in PPM architecture

9.3.2.1 Join to a M2M platform

When a data subject joins an M2M platform, the data subject configures a privacy preference using the PPM. A privacy preference explains what kind of data are allowed to access ASPs. Figure 9.3.2.1-1 illustrates the overview of this process.

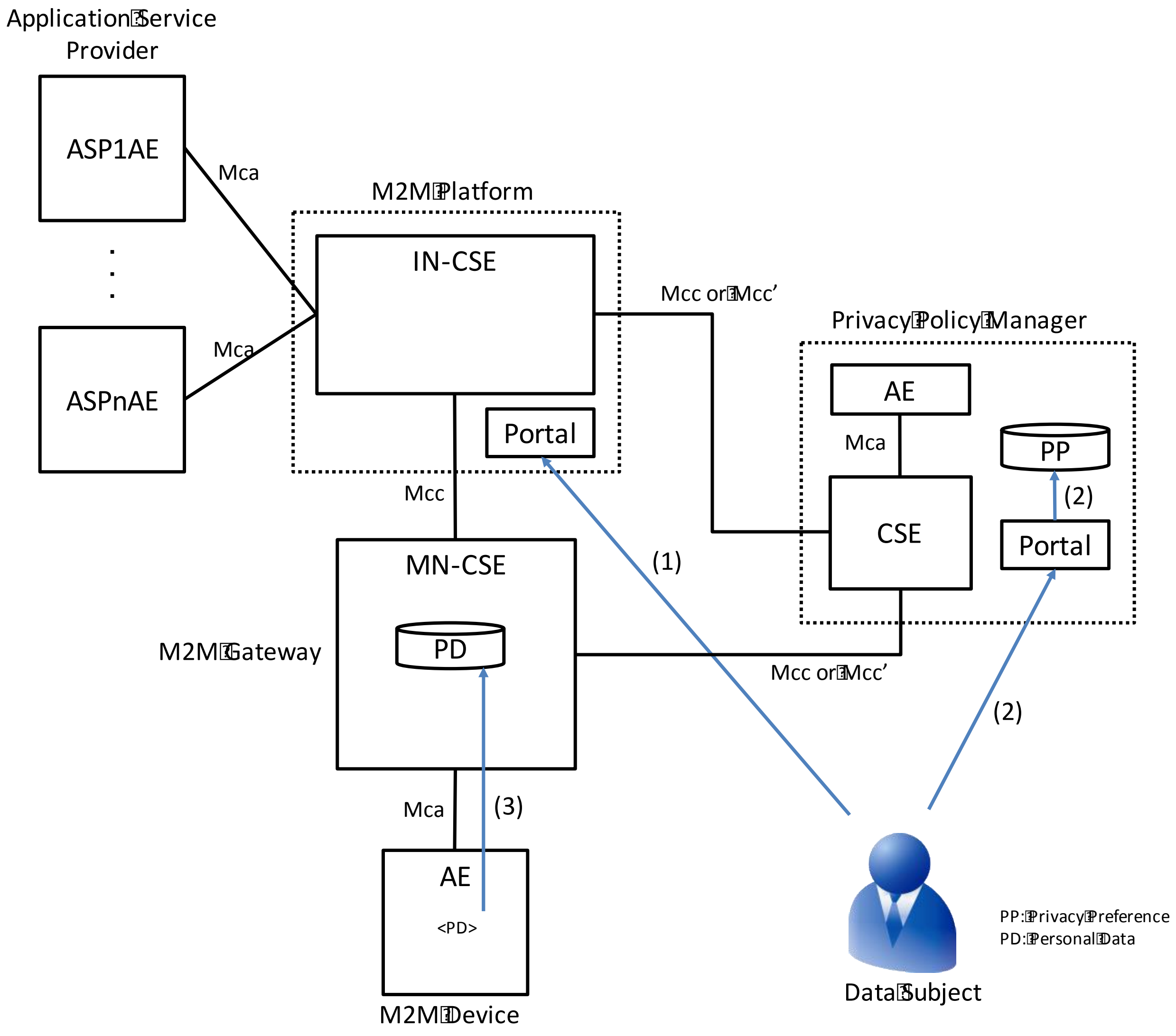

Figure 9.3.2.1-1: A data subject join a M2M Platform

1) A data subject accesses a portal on an M2M platform:
   - This process is equivalent to protocols of Web access such as HTTP, HTTPS and so on.

- This prosess is for further study of oneM2M.

2) A data subject configures a privacy preference and registers it on the PPM. Then, the PPM creates acccess control policies based on it:

   - A data subject accesess the PPM or the M2M platform redirects the data subject to the PPM. This process is equivalent to protocols of Web access.

   - This process is for further study of oneM2M.

3) The M2M gateway collects and stores data from M2M devices.

9.3.2.2 Subscription to an ASP's service

The data subject can subscribe to various kinds of services provided by ASPs through the M2M platform. Service lists are registered in the M2M platform and the data subject can select services to subscribe to. When the data subject subscribes to a service, the data subject needs to accept a privacy policy. In order for the data subject to easily understand, the PPM creates the customized privacy policy based on the ASP's privacy policy and the data subject's privacy preference. Therefore, the data subject can control personal data and prevents agreement without understanding the privacy policy. Figure 9.3.2.2-1 shows the overview of this process.

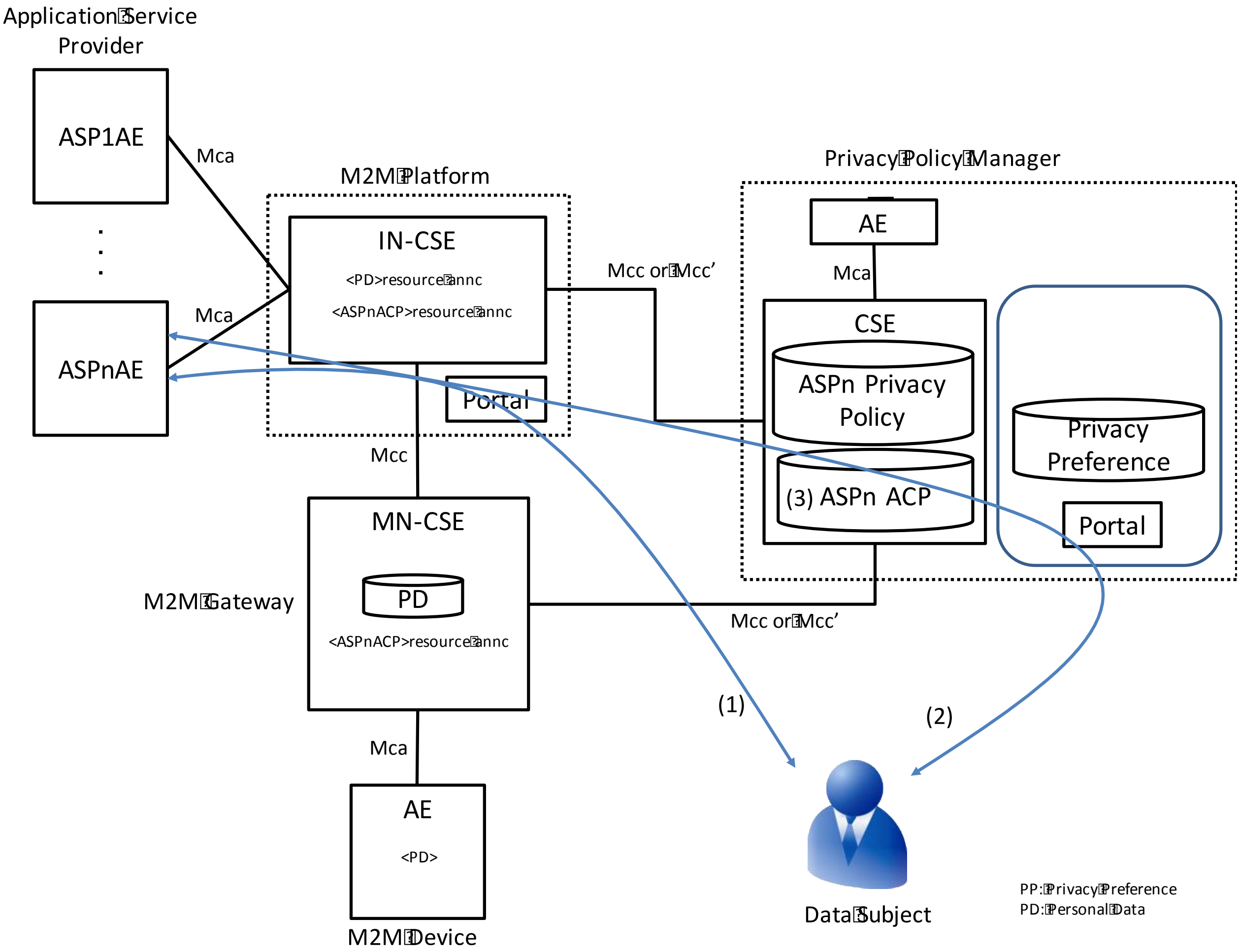

Figure 9.3.2.2-1: The data subject subscribes to an ASP's service

1) The data subject accesses the portal and select an ASP's service to subscribe:

   - This process is equivalent to protocols of Web access such as HTTP, HTTPS and so on.

   - This process is for further study of oneM2M.

2) The data subject needs to accept a privacy policy to subscribe to the ASP's service. The PPM creates the customized privacy policy for each data subject based on the data subject's privacy preference. It is easy for the data subject to confirm differences between the privacy preference and the privacy policy and to understand what kind of personal data are collected by the ASP. After the data subject accepts the privacy policy, the data subject can subscribe to the ASP's service:

   - The function of creating a customized privacy policy is for further study of oneM2M.

3) The PPM creates or updates access control policies using the privacy policy that the data subject accepted:

   - The function of creating or updating access control policies in the PPM is for further study of oneM2M.

9.3.2.3 Request for personal data to the M2M platform

If the ASP needs personal data to provide the service, the ASP requests the data from the M2M platform. The PPM can work as PDP or PRP. If the PPM works as PDP, a data flow control recieves access decision from the PPM and controls the data access using them. Figure 9.3.2.3-1illustrates the overview of this process. If the PPM works as PRP, the data flow control retrieves access control policies from the PPM and controls the data access using them. Figure 9.3.2.3-2 illustrates the overview of this process.

A. The PPM works as PDP

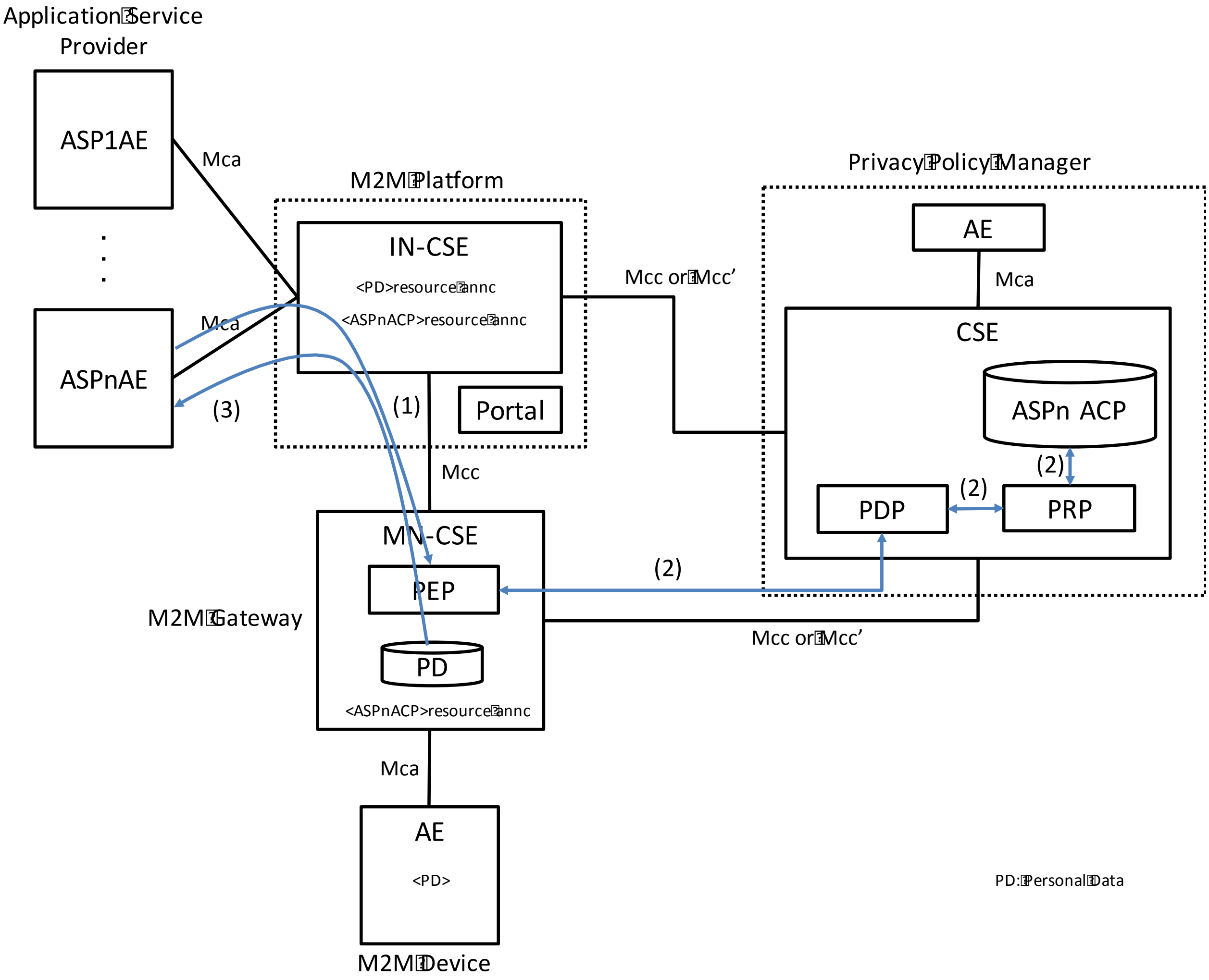

Figure 9.3.2.3-1: Request for personal data to the M2M platform (the PPM works as PDP)

1) ASP requests personal data from the M2M platform.

2) PEP in the M2M gateway requests "Decision Request" from the PPM. The PPM decides to permit/deny access to personal data using access control policies. Then, the PPM replies "Decision Response".

- In this case, the PPM needs to provide an interface that enables access control for personal data using the PPM as PDP.

3) If accessing personal data is permitted, PEP accesses the personal data and sends the personal data to ASP as response.

B. The PPM works as PRP

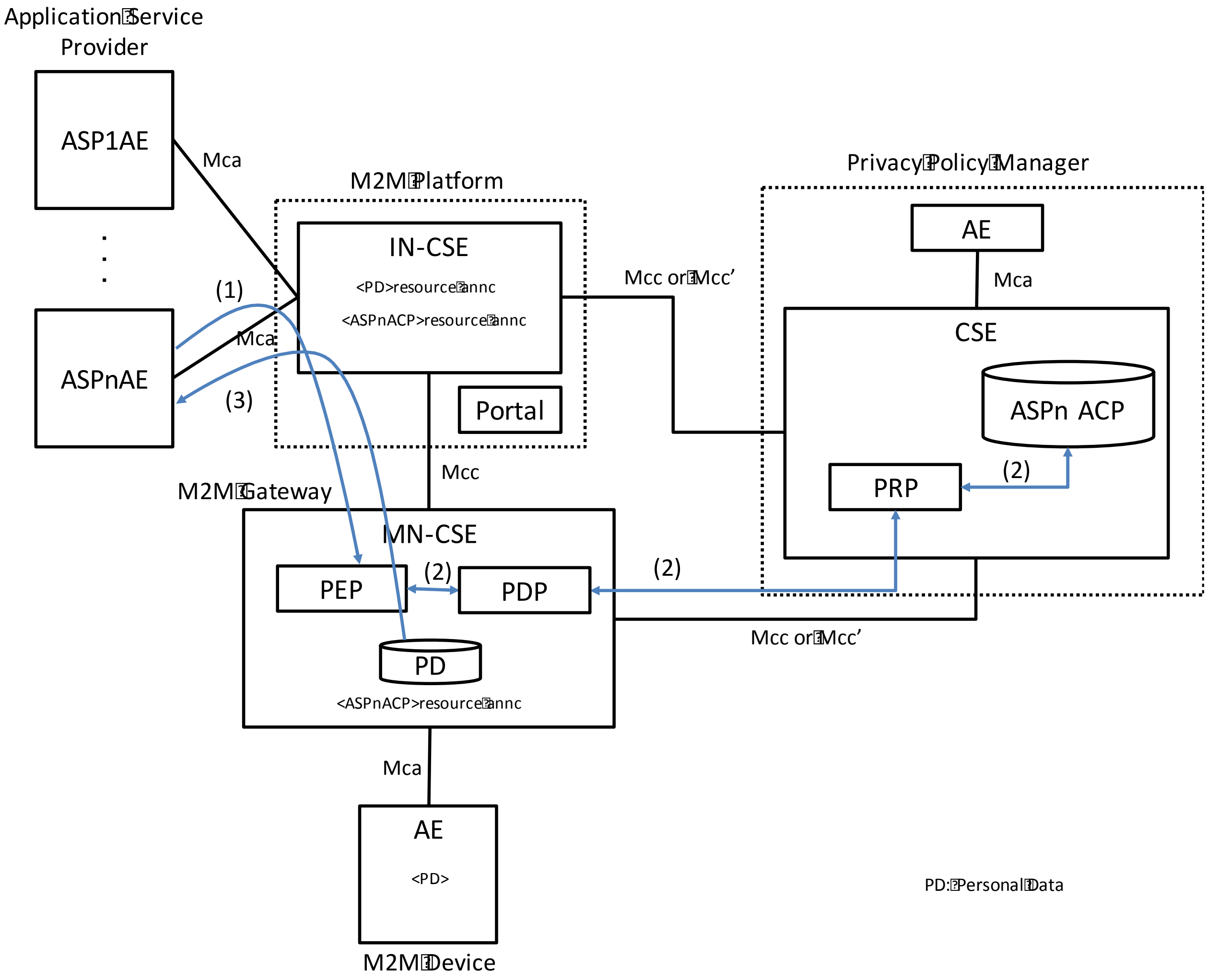

Figure 9.3.2.3-2: Request for personal data to the M2M platform (the PPM works as PRP)

1) ASP requests personal data from the M2M platform.

2) PEP in the M2M gateway requests "Decision Request" from PDP. PDP requests "Policy Request" from the PPM. The PPM collects access control policies about "Policy Request" and sends the access control policies as "Policy Response" to PDP in the M2M platform. Then, PDP decides to permit or deny access to the personal data using the access contror policies and sends a result as "Decision Response" to PEP:

   - In this case, the PPM needs to provide an interface that enables access control for personal data using the PPM as PRP.

3) If accessing personal data is permitted, PEP accesses the personal data and sends the personal data to the ASP as response.

10 Conclusions

The present document focuses on the authorization architecture, access control polices and privacy policy management in the oneM2M System. The investigated topics and corresponding conclusions are summarized as follows:

- Clause 6.2 proposes distributed authorization architecture and a way of using virtual resources to implement distributed authorization system. It is also possible for oneM2M using normal resources to implement distributed authorization. Compared with virtual resource based solution, normal resource based solution is more in line with RESTful API. Currently both ways are being evaluated by oneM2M Technical Specifications.

- Clause 6.3 investigates the possibility of using XACML and SAML to exchange authorization information between authorization components. These technologies are powerful for expressing authorization information, but for oneM2M it is too powerful and overburdened. So they will not be considered inner oneM2M system. However, these technologies may be considered when there is authorization information exchanging with external systems. Currently the way of using request and response parameters to exchange authorization information between authorization components is being developed.

- Clause 6.4 provides a role based access control solution including RBAC architecture and generic procedures. The basic idea (e.g. RBAC architecture, <*role*> resource type and procedures) proposed in the present document has been integrated into the Release 2 of the oneM2M Technical Specifications.

- Clause 6.5 investigates some potential attribute based access control solutions. A generic ABAC procedure is also proposed. However, these solutions are not mature. How to implement ABAC in the oneM2M System needs further study. It may require a special work item to study this topic.

- Clause 7 provides a solution of supporting user specified access control polices. The proposed solution can also be used to support privacy policy enforcement. Currently, as the normative work is not yet started, it will not be part of the Release 2 of oneM2M Technical Specifications. However, such feature is quite useful for the oneM2M System, as the access control of resources in the oneM2M System may involves multiple stakeholders. So this feature may be considered by the future release, and the solution proposed in the present document may be used to facilitate future normative work resulting in oneM2M Technical Specifications.

- Clause 8 investigates some existing access control policy languages in the context of the oneM2M System. XACML is the only available ABAC access control policy language. Advantages and disadvantages of XACML have been enumerated. If the oneM2M System wants to consider XACML as one of its ABAC policy languages, a special work item may be required for further study. Some limitation of current oneM2M access control policy is identified and a solution is proposed. Some part of the proposed solution has been integrated into the Release 2 of the oneM2M Technical Specifications.

- Clause 9 investigates how the privacy protection could be considered in the oneM2M System. A framework named privacy policy manager is introduced. This clause describes how the privacy policy manager and the oneM2M authorization system work together to implement privacy protection. The proposed solution has been integrated into the Release 2 of TS-0003.

History

Publication history		
V2.0.0	30-Aug-2016	Publication