|  |  |
| --- | --- |
| Input Contribution | |
| Meeting ID\* | TST#34 |
| Title:\* | Input on “Authorization” procedure for the Security Developer’s Guide TR-0038 |
| Source:\* | Wolfgang Granzow, Qualcomm, [wgranzow@qti.qualcomm.com](mailto:wgranzow@qti.qualcomm.com)  Phil Hawkes, Qualcomm, [phawkes@qti.qualcomm.com](mailto:phawkes@qti.qualcomm.com) |
| Uploaded Date:\* | 2018-03-13 |
| Document(s)  Impacted\* | TR-0038v0\_4\_0 |
| Intended purpose of  document:\* | Decision  Discussion  Information  Other <specify> |
| Decision requested or recommendation:\* | Include the text proposed in this contribution into TR-0038v0.3.0 |
| Template Version:23 February 2015 (Do not modify) | |

**oneM2M Notice**

The document to which this cover statement is attached is submitted to oneM2M. Participation in, or attendance at, any activity of oneM2M, constitutes acceptance of and agreement to be bound by terms of the Working Procedures and the Partnership Agreement, including the Intellectual Property Rights (IPR) Principles Governing oneM2M Work found in Annex 1 of the Partnership Agreement.

**Introduction**

This contribution proposes text for the currently empty sections of TR-0038:

* Section 7.2 “Authorization” is addressed in detail.
* Section 7.3 on “Secure communications”
* Section 8 “Implementation” is proposed to be removed since all implementation specific aspects are covered in the previous sections and in the Annexes.
* Text for section 9 “Conclusions” is proposed to be removed. Such section doesn’t exist in any other Application developer guide.

*======== start of change 1 ===========*

## 7.2 Authorisation

### 7.2.1 Introduction

The Authorization function is responsible for controlling access to resources and services hosted by CSEs and AEs.

The authorization procedure requires that the originator of the resource access request message has been identified to the Authentication function, and originator and receiver are mutually authenticated with each other. Mutual authentication between adjacent entities, i.e. between registree and registrar, can be ensured by the Security Association Establishment procedures as described in clause 7.1.

In the oneM2M system, access to resources can be controlled by assignment of access control policies to the resources. Access control policies govern *who* (originators) can do *what* (operations) under *which* circumstances (context information associated with a request).

Access control policies can be configured in the form of <*accessControlPolicy*> resources (ACP) which are statically assigned to other resources by means of an *accessControlPolicyID* attribute. The *accessControlPolicyID* attribute can include a list of resource identifiers of <*accessControlPolicy*> resources which include the access control rules applicable to that resource. This is illustrated in Figure 7.2.1-1. The links refer to the elements included in the *accessControlPolicyID* attribute. Each configured <*accessControlPolicy*> resource ACP1…3 includes one or more ACP rule(s). Each such ACP rule *who* can do *what* under *which* circumstances.

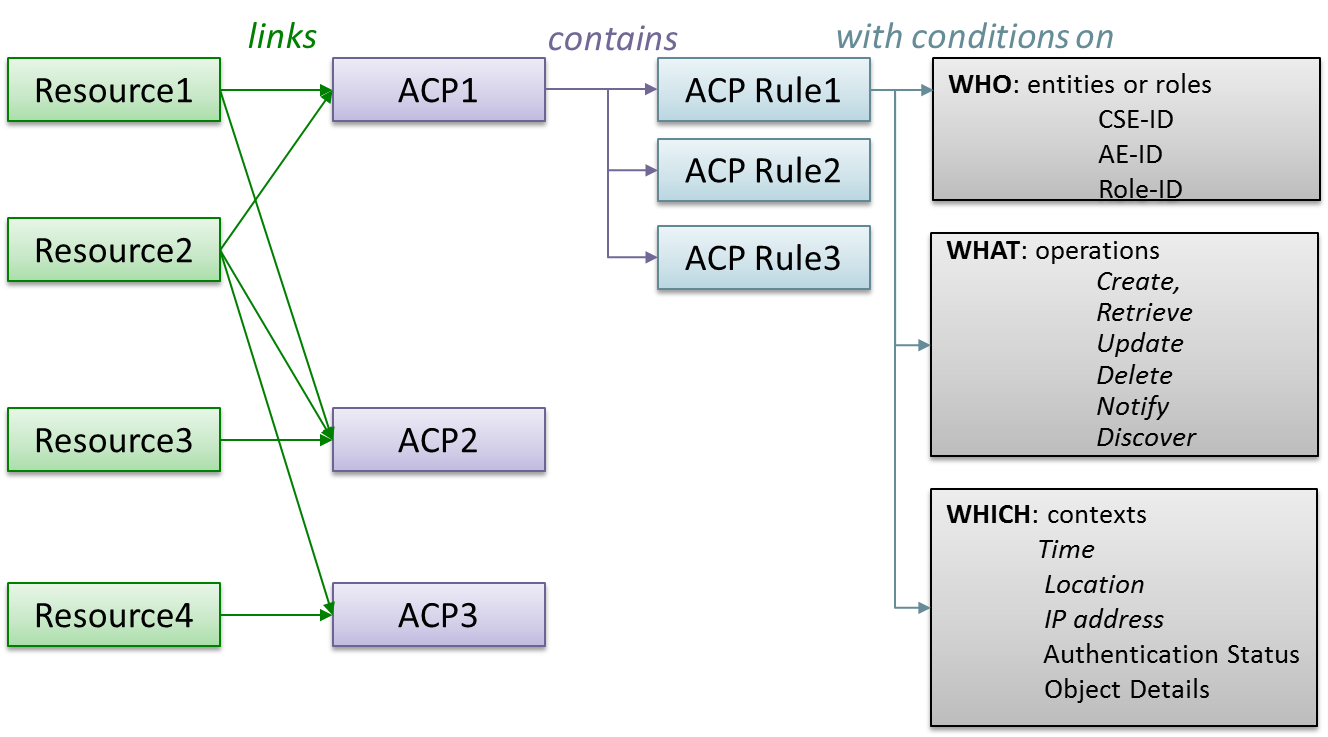


Figure 7.2.1-1: Assignment of Access Control Policies (ACP) to resources

The details of access control policy information and the access control mechanism are specified clause 7.1. of TS-0003.

This clause focuses on a simple example of configuring access control policy information adequate for the considered use case.

More advanced access control mechanisms, which employ dynamic access control, role-based access control and distributed access control are not addressed in the present version of this document.

### 7.2.2 Resource structure of the example use case

Figure 7.2.2-1 shows an example resource tree hosted by the MN-CSE which is suitable for the door lock use case as described in clauses 5 and 6.

The <*AE*> resources representing the two door locks are created at registration. The resource tree under each of these <*AE*> resources should look the same. Therefore, the figure exemplifies only the resource structure under ADN-AE1. After completion of the <AE> registration procedure it is assumed that following procedures are executed by each door lock:

1. Creation of a <*container*> resource representing the state information of the respective door lock;
2. Creation of a first <*contentInstance*> resource, which includes the actual door lock state (i.e. “locked” or “unlocked”) in the *content* attribute, e.g. in the form of a binary representation;
3. Creation of a <*pollingChannel*> resource to be employed by the door lock AE;
4. Creation of a <subscription> resource under the <container> resource, which defines conditions for which a notification is sent to the respective door lock application;
5. Creation of another <*subscription*> resource which defines conditions for which a notification is sent to the door lock controller application. This resource is created by the door lock controller (see below).

Note that the detailed procedures to create the above resources are not in the scope of the present document. These procedures are described in the Applications Developer Guide TR-0025.



Figure 7.2.2-1: Resource tree hosted by the MN-CSE

The door lock controller ADN-AE3 implemented on the smartphone registers to the IN-CSE. The created <*AE*> resource does not require <container> child resources for its function. We assume in this example that ADN-AE3 is not request reachable and therefore requires also a <*pollingChannel*> child resource. In this case, after completion of the <AE> registration procedure, ADN-AE3 is assumed to execute following procedures:

1. Creation of a <*pollingChannel*> resource to be employed to retrieve the <*pollingChannelURI*> virtual child resource;
2. Creation of the <*subscription*> resource under each of the <*container*> resources of ADN-AE1 and ADN-AE2. This <*subscription*> resource defines conditions for which a notification is sent to the door lock controller ADN-AE3.

### 7.2.3 Configuration of <*accessControlPolicy*> resources

The resource types defined by the oneM2M specifications can be broadly categorized into two classes:

1. Resource types which have an optional *accessControlPolicyID* attribute. These are denoted as “regular resource types” in the following (cf. clause 6.5 of TS-0004).
2. Resource types which do not have an optional *accessControlPolicyID* attribute. These are denoted as “subordinate resource types” in the following (cf. clause 6.5 of TS-0004).

Access control to subordinate resource types is specified on a case-by-case basis for each individual resource type in TS-0001. The <*accessControlPolicy*> and <*pollingChannel*> belong into this category.

Resources of type <*accessControlPolicy*> include a *selfPrivileges* attribute which defines access privileges to change an <*accessControlPolicy*> itself.

Resources of type <*pollingChannel*> are accessible by the creator of each resource instance only.

For “regular resource types” which do not have any *accessControlPolicyID* attribute assigned yet, default access privileges apply. The default access privilege gives the creator unrestricted access to the resource, i.e. it allows the creator of the resource to execute all possible operations defined for that resource type.

Access control management of “regular resource types” generally consists of two steps:

1. Creation of suitable <accessControlPolicy> resources
2. Setting of the *accessControlPolicyID* attribute in applicable resources

When an <*AE*> resources is created at AE registration, access control policies do not apply. Authorization is done solely based on M2M service subscription information, as outlined in clause 7.1.5.

Thanks to the default access privilege, the originator/creator of the <AE> resource is allowed to create child resources as well as children of children. This means, the resource tree shown in Figure 7.2.2-1 under the <*AE*> resource of ADN-AE1 or ADN-AE2 can be created without any <*accessControlPolicy*> resources assigned in the *accessControlPolicyID* attribute.

However, when originators other than the creator of the <*AE*> resource should be given access, then access control policies must be assigned. For the use case example considered here, at least access control policies must be configured which allow the door lock controller ADN-AE3 to update and retrieve the <container> resources created by the door lock applications ADN-AE1 and ADN-AE2 and to create a <subscription> to these containers.

An <*accessControlPolicy*> resource contains two mandatory resource-specific attributes, denoted *privileges* and *selfPrivileges*. Each of these attributes includes one or more *access control rule(s)*. An access control rule has two mandatory elements, namely *accessControlOriginators* and *accessControlOperations*. In addition, there can be up to three optional elements, denoted *accessControlContexts*, *accessControlAuthenticationFlags*, and *accessControlObjectDetails*.

We focus on the mandatory elements of an access control rule first. The *accessControlOriginators* element of an access control rule represents a list of originators (i.e. AE-IDs or CSE-IDs) which are allowed to perform operations defined in the *accessControlOperations* element. See clause 7.1.3 and Table 7.1.3-1 in TS-0003 for a detailed description of the elements of access control rules. TS-0004 defines how the values of elements and sub-elements are represented in terms of XML schema datatypes.

There are several implementation options how to setup <*accessControlPolicies*> in a oneM2M system.

If these resources are to be created and managed in a standard compliant way, the natural approach is to employ an AE for this purpose. This could be a special AE just serving the purpose of managing access control, or it could be implemented as an additional function of an AE which also serves other purposes.

The following design options may be considered for the considered door lock use case

1. Develop a separate AE which registers to the MN-CSE directly. This could be either a separate ADN-AE or an MN-AE, i.e. an AE residing on the same device as the MN-CSE.
2. Develop a separate AE which registers to the IN-CSE and which can access the MN-CSE. In this case it could be implemented either as integral part of the door lock controller ADN-AE3 or it could be implemented as a separate additional application which runs on the same ADN (smartphone) as ADN-AE3.
3. The AE employed for setting of access control policies is an IN-AE managed by an M2M service provider. In this case management of access control policies is executed under responsibility of the M2M service provider based on some agreement between the end user and service provider.
4. The AE may function in a fully automated manner or in a semi-automated manner requiring manual interaction by a human user. If the latter case is desired, the device hosting the AE must have capability to provide a rich graphical user interface (e.g. personal computer or a smart phone)

In the following we consider the implementation of an AE (denoted AE4) which exclusively serves configuration of access control policies. Such AE could be deployed flexibly on different M2M devices in accordance with a user’s preference.

For the considered door lock use case, the AE should provide the following basic functionality:

1. Discovery of any AEs associated with the given service
2. Interpretation of the function of each discovered AE (e.g. from App-ID)
3. Creation of <accessControlPolicies> resources on the MN-CSE
4. Setting of the *accessControlPolicyID* attribute

An example representation of the *privileges* and *selfPrivileges* attributes equivalent with what is denoted as “default access privilege” to resources created by C-lock-AE1:

<privileges>  
 <accessControlRule>  
 <accessControlOriginators>C-lock-AE1</accessControlOriginators>  
 <accessControlOperations>63</accessControlOperations>   
 </accessControlRule>  
 </privileges>  
 <selfPrivileges>  
 <accessControlRule>  
 <accessControlOriginators>C-lock-AE1</accessControlOriginators>  
 <accessControlOperations>63</accessControlOperations>  
 </selfPrivileges>

Note that once access privileges are assigned to a resource in the *accessControlPolicyID* attribute, the default access rule does not apply anymore. If the default access privilege should remain in place it needs to be defined explicitly and made part of the applicable set of access control rule (either as a separate <accessControlPolicy> resource, or as a specific access control rule which is included with other rules into an<accessControlPolicy> resource.

The *accessControlOriginators* element of an access control rule is represented as a list of members which can a type as given in table 7.2.3-1.

Table 7.2.3-1: Types of *accessControlOriginators* element

| Member Type | Criterion to pass this constraint |
| --- | --- |
| *SP Domain name* | FQDN of a service provider’s domain, e.g. area10023.myprovider.org. All AEs and CSEs in this domain are granted access within the *accessControlOriginators* constraints |
| *originatorID* | a) CSE-ID, AE-ID, wildcard character ‘\*’ allowed.  b) resource-ID of a <group> resource that contains the AE or CSE representing the originator, no wildcard allowed.  Originator of the request which matches the given CSE-ID or AE-ID is granted access within the *accessControlOriginators* constraints |
| *Key word “all”* | Any Originators are allowed to access the resource within the *accessControlOriginators* constraints |
| *Role-ID* | 1. Role Identifier associated with an AE /AE-ID as defined in *allowedRole-ID* attribute of <*serviceSubscribedAppRule*> 2. Role identifier associated with an AE /AE-ID as defined in a <role> resource   Example Role-ID: [1234abcd@role-issuer.com](mailto:1234abcd@role-issuer.com) |

The *accessControlOperations* element of an access control rule is represented as decimal number in the range of 1 … 63 which represents an encoded combination of permitted operations on the resource. The encoding is defined in table 7.2.3-1.

When converting the decimal number into a 6-bit binary representation, each binary digit corresponds to one specific operation as illustrated in Table 7.2.3-2. A digit with value 1 or 0 means that the respective operation is allowed or disallowed, respectively. In other words, the digit “1” represents a flag that the corresponding operation is permitted.

Table 7.2.3-2: Representation of accessControlOperations parameter

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Enumeration | Discov. | Notify | Delete | Update | Retrieve | Create |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 0 | 1 | 1 |
| … | … | .. | … | … | … | … |
| 63 | 1 | 1 | 1 | 1 | 1 | 1 |

For example, if CRUD operations should be allowed and Notify and Discovery disallowed, the value of *accessControlOperations* parameter needs to be set to 15 (binary: 001111).

CRUD and Discovery represent operations which are executed on the resource addressed in the ***To*** parameter of a request primitive. A Notify request message, however, does not represent an operation on a resource.

A Notify request message (aka. Notification) is typically sent to an entity (AE or CSE) to inform it, that a special event has occurred which the receiver of the Notification has subscribed to by means of a <subscription> resource.

Other use case for Notifications include the transfer of the response primitive in reply to a request which is sent in non-blocking asynchronous transmission mode and the response to long polling (i.e. Retrieve request targeting at a <pollingChannelURI> virtual resource).

See clause 7.5.1.2 of TS-0004 [1.3] for a comprehensive description of Notification use cases.

Notify request primitives are sent to the entity which is identified by the ***To*** parameter and denoted as notification target. Notifications which are triggered by conditions defined in a <subscription> resource are sent to the notification target(s) given in the *notificationURI* attribute of the <subscription> resource. *notificationURI* attribute is represented as a list which can include one or more members. The applicable formats of each member of this attribute are specified in clause 9.6.8 of TS-0001 [i.2].

Notification targets are represented as a oneM2M resource-ID which can be represented in various formats as defined in clause 7.2 of TS-0001 [i.2].

The Notify “flag” in the *accessControlOriginators* element is validated for every Notify request message sent to either an AE or CSE. The notification target, i.e. the ***To*** parameter of a Notify request primitive is represented in the form of a resource-ID of an <AE> or <remoteCSE> (<CSEBase>??) resource. The Notify “flag” in the *accessControlOriginators* element of the <AE> or <remoteCSE> associated with the notification target must be set to pass this access control condition. The Notify “flag” indicates that the respective entity is allowed to receive Notify request messages.

*Editor’s Note: TS-0001 currently seems to allow to send Notifications to non-oneM2M entities provided these are compliant with any of the oneM2M defined protocol bindings. This however may impose a security risk since there is no guarantee that the non-oneM2M entity complies with oneM2M security measures. The SEC WG should discuss this and possible security threats jointly with the ARC WG. This feature should possibly be removed from the specifications.*

*Editor’s Note:*

To be added in the next update of this contribution:

* Details of configuration and assignment of ACPs to the resources shown in Figure 7.2.2-1, i.e. ACP assigned to <AE>, <container>, <subscription>

## 7.3 Secure communications

Once a security association is established between adjacent oneM2M nodes, all communication between these nodes is secured. However, all data of request and response messages is visible in the clear to both end points of a security association. Messages which need to be forwarded by an MN-CSE or IN-CSE are re-encrypted using the security context established with the next-hop node. Any intermediate CSE must be trusted in this communication scenario. If a communication path may include CSEs which cannot be trusted, end-to-end security mechanisms need to be employed.

The present version of this document focuses on secure communication between adjacent nodes. Future versions will also address examples of configuring end-to-end communication using the ESPrim and ESData mechanisms specified in TS-0003.

*======== End of change 1 =============================*