|  |
| --- |
|  |

|  |  |
| --- | --- |
| CHANGE REQUEST | |
| Meeting ID:\* | SDS #50 |
| Source:\* | Andreas Kraft, DT, [Andreas.Kraft@t-systems.com](mailto:Andreas.Kraft@t-systems.com)  Andreas Neubacher, DT, [Andreas.Neubacher@magenta.at](mailto:Andreas.Neubacher@magenta.at) |
| Date:\* | 2021-04-22 |
| Reason for Change/s:\* | Editorial corrections for TS-0003 (revised) |
| CR against: Release\* | Release 4 |
| CR against: WI\* | Active WI-xxxx  MNT maintenance / < Work Item number(optional)>  Is this a mirror CR? Yes  No  mirror CR number: (Note to Rapporteur - use latest agreed revision)  STE Small Technical Enhancements / < Work Item number (optional)>  Only ONE of the above shall be ticked |
| CR against: TS/TR\* | TS-0003, V4.4.0 |
| Clauses \* | 7.1 |
| Type of change: \* | Editorial change  Bug Fix or Correction  Change to existing feature or functionality  New feature or functionality  Only ONE of the above shall be ticked |
| Impacted other TS/TR(s) |  |
| Post Freeze checking:\* | This CR contains only essential changes and corrections? YES  NO  This CR may break backwards compatibility with the last approved version of the TS? YES  NO |
| Template Version: January 2017 (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.

GUIDELINES for Change Requests:

Provide an informative introduction containing the problem(s) being solved, and a summary list of proposals.

Each CR should contain changes related to only one particular issue/problem.

In case of a correction, and the change apply to previous releases, a separate “mirror CR” should be posted at the same time of this CR

Mirror CR: applies only when the text, including clause numbering are exactly the same.

Companion CR: applies when the change means the same but the baselines differ in some way (e.g. clause number).

Follow the principle of completeness, where all changes related to the issue or problem within a deliverable are simultaneously proposed to be made E.g. A change impacting 5 tables should not only include a proposal to change only 3 tables. Includes any changes to references, definitions, and acronyms in the same deliverable.

Follow the drafting rules.

All pictures must be editable.

Check spelling and grammar to the extent practicable.

Use Change bars for modifications.

The change should include the current and surrounding clauses to clearly show where a change is located and to provide technical context of the proposed change. Additions of complete clauses need not show surrounding clauses as long as the proposed clause number clearly shows where the new clause is proposed to be located.

Multiple changes in a single CR shall be clearly separated by horizontal lines with embedded text such as, start of change 1, end of change 1, start of new clause, end of new clause.

When subsequent changes are made to content of a CR, then the accepted version should not show changes over changes. The accepted version of the CR should only show changes relative to the baseline approved text.

Introduction

This CR proposes a couple of editorial corrections for TS-0003.

After discussions which happened after the CR SDS-2021-031R01 was agreed, it was discussed and agreed NOT to change the attribute’s element name “accessControlWindow”.

Change all occurrences of “accessControl**Time**Window” to “accessControlWindow” in Change 1

### \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Start of Change 1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## 7.1 Access Control Mechanism

### 7.1.1 General Description

The M2M authorization procedure controls 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.

The resource addressed in a request message has an associated *accessControlPolicyIDs* attribute (either included explicitly as an attribute of the resource addressed in the request message, implied from the parent of the resource, or set fixed by the system, see clause 9.6.1 of oneM2M TS-0001 [1]). The *accessControlPolicyIDs* attribute contains a list of identifiers of <*accessControlPolicy*> resources applicable to the resource addressed in the request message.

The overall structure of <*accessControlPolicy*> resources is described in clause 9.6.2 "Resource Type *accessControlPolicy"* of oneM2M TS-0001 [1]).

Each of these <*accessControlPolicy*> resources include *privileges* and *selfPrivileges* attributes, which comprise the information, denoted as *access control rules* in the present document, that is evaluated against the parameters associated with the request message to obtain the access decision.

Figure 7.1.1-1 illustrates the relation between <*accessControlPolicy*> resource instances (ACP) and the instances of the protected resources, denoted Resource\_1 to Resource\_N.

Resource\_2

**...**

ACP\_1

ACP\_2

ACP\_3

ACP\_4

Instances of

accessControlPolicy

resources (ACP)

*Example:*

ACP set = (ACP\_1, ACP\_2)

assigned to Resource\_1

Each ACP includes one *privileges* and one *selfPrivileges* attribute.

*privileges* and *selfPrivileges* attributes include a set of access control rules (defined in Section 7.3)

List of IDs in accessControlPolicyIDs

attribute of Resource\_1

Resource\_1

Resource\_3

Resource\_N

Figure 7.1.1-1: Relation between Resource Instances and Access Control Policies

Access requests to ACP's itself are evaluated against the *selfPrivileges* attribute of that ACP. Access requests to instances of all other resource types, are evaluated against the *privileges* attributes of the ACP set associated with the targeted resource.

For requests to <*accessControlPolicy*> resource type, authorization is granted if the request is evaluated to "Permit" for at least one *selfPrivileges* attribute. For other resource types, authorization is granted if the request is evaluated to "Permit" for at least one *privileges* attribute.

The *privileges* and *selfPrivileges* defined in the *accessControlPolicy* resource determine *which* *request originator* is allowed to access the resource containing this attribute, for *which specific operation* (i.e. Create, Retrieve, Update, Delete, etc.) and *for which specific context constraints* (i.e. constraints regarding access time, originator's IP address and originator's location).

The access control approach specified here conforms to the concept of Attribute Based Access Control (ABAC) as defined in [i.12].

The policies defined in the <*accessControlPolicy*> resources are enforced by an access control mechanism which employs the authorization logical architecture outlined in clause 6.2.2.

The access control mechanism assembles the information needed to render the access decision which consists of:

* Information included in the resource access request message as defined in clause 7.1.2 (table 7.1.2‑1).
* Contextual information as defined in clause 7.1.2 (table 7.1.2-2).
* Tokens (if any) associated with the resource access request.
* The policies governing the access as defined in clause 7.1.3.

### 7.1.2 Parameters of the Request message

This clause specifies the parameters of a request message which are evaluated by the access control mechanism.

The data types applicable to these parameters are defined in clause 6.4 of oneM2M TS-0004 [4].

The parameters are listed in table 7.1.2-1.

Table 7.1.2-1: Parameters indicated in the request message

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Description | Mandatory/ Optional | Usage in access control mechanism |
| *To* | URI of target resource | M | Selection of accessControlPolicy associated with the target resource |
| *From* | Identifier representing the originator of the request | M  (see note 1) | Evaluated against accessControlOriginators in *privileges* and *selfPrivileges* attributes |
| *Role IDs* | Role IDs of the originator | O | Evaluated against accessControlOriginators in *privileges* and *selfPrivileges* attributes |
| *Operation* | Requested operation | M | Evaluated against accessControlOperations in *privileges* and *selfPrivileges* attributes |
| *Resource Type* | Type of the target resource | O (see note 2) | Evaluated against accessControlObjectDetails in *privileges* attributes. Applicable to Create operations only. |
| *Filter Criteria* | ***filterUsage*** condition tag in Filter criteria | O | Differentiation between Retrieve and Discovery operations |
| *Tokens* | ESData-protected Tokens | O | Contains authorization information (e.g. Role-IDs) to be used in the decision for the request |
| *Token IDs* | tokenIDs **or** Local-Token-ID | O | Identifies Tokens containing authorization information (e.g. Role-IDs) to be used in the decision for the request |
| *M2M Service User* | Identity of a M2M Service User | O | Evaluated against the accessControlUserIDs sub-parameter of the accessControlContexts parameter of the *privileges* and *selfPrivileges* attributes |
| NOTE 1: The ***From*** primitive parameter is Mandatory in all requests except for AE registration procedure where it is optional, as specified in oneM2M TS-0001 [1].  NOTE 2: The ***resource Type*** primitive parameter is present in Create request primitives only. | | | |

Table 7.1.2-2 lists the context parameters associated with a request message which are evaluated by the access control mechanism. These parameters are not explicitly included in a request message but can be obtained at the receiver and validated against the context policy parameters as given in table 7.1.2-2.

Table 7.1.2-2: Context parameters associated with a request message

|  |  |  |
| --- | --- | --- |
| Parameter | Description | Usage in access control mechanism |
| *rq\_time* | Time stamp when the request message was received at the hosting CSE. Obtained by the hosting CSE's system time clock. | Validated against accessControlWindow parameter in an access control rule, see clause 7.1.3. |
| *rq\_loc* | Location information about the originator of the request. Obtained over the Mcn reference point. | Validated against accessControlLocationRegion parameter in an access control rule, see clause 7.1.3. |
| *rq\_ip* | IP source address associated with the IP packets that carry the request message. Obtained over the Mcn reference point. | Validated against accessControlIpAddresses parameter in an access control rule, see clause 7.1.3. |

Tokens, as defined in clause 7.3.3.1 "Token Structure", may be associated with a request message. A Token may be associated with a request as a result of being included in the ***Tokens*** primitive parameter of the request message or identified in the ***Token IDs*** primitive parameter of the request message. If the Hosting CSE obtained a token from the Dynamic Authorization System (DAS) Server using Direct Dynamic Authorization, then this Token shall be associated with a request if the holder parameter in the Token matches the Absolute AE-ID or CSE-ID of the Originator of the request. Dynamic Authorization is specified in clause 7.3.

Table 7.1.2-3 lists the security context parameters associated with a request message.

Table 7.1.2-3: Security Context parameters associated with a request message

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Description | Mandatory/Optional | Usage in access control mechanism |
| *rq\_authn* | Boolean value (TRUE/FALSE) indicating if the Originator is considered to have been authenticated by the Hosting CSE, and the ***From*** parameter matched the authenticated identity of the Originator. | M | Validated against accessControlAuthenticationFlag parameter in an access control rule, see clause 7.1.3. |

The following criteria shall be applied to determine if an Originator is considered to have been authenticated by the Hosting CSE.

* If the Originator is an AE registered to the Hosting CSE, then the criteria for deciding whether the Originator is considered authenticated is deployment and/or implementation specific and depends on the trust guaranteed by the field device's physical and logical embodiment bearing the AE(s) and Hosting CSE (e.g. secure boot and tamper resistance). In many cases it is appropriate to expect a secure channel implying authentication (e.g. a TLS or DTLS session) to be used to protect primitives on the Mca interface, in which case the authentication shall be considered valid for the duration of the TLS session, When this is not the case, e.g. because the physical and logical design is trusted, authentication may be considered to be permanently valid unless it is detected that the device is compromised.
* If the Originator is a CSE registered with the Hosting CSE, then the Originator shall be considered authenticated for the duration of a (D)TLS session because the Mcc is always required to be protected by TLS or DTLS according to a Security Association Establishment Framework (SAEF) as described in clause 8.2. The other CSE may be the Registrar or Registree with respect to the Hosting CSE.
* If the Originator is an AE or CSE registered with a CSE other than the Hosting CSE, then the Originator is considered authenticated by the Hosting CSE if and only if the request primitive is protected using End-to-End Security of Primitives (ESPrim) as described in clause 8.4.

### 7.1.3 Format of *privileges* and *selfPrivileges* Attributes

The *privileges* and *selfPrivileges* attributes exhibit the same data type format which is specified as follows.

Each *privileges* or *selfPrivileges* attribute comprises a set of access control rules. In the following, the set of access control rules is denoted as *acrs* and an individual access control rule in this set as *acr*. The access control rules in *acrs* are indexed with the letter *k*. The number of access control rules in the set is denoted with the letter K:

*acrs* = { *acr*(1), *acr*(2), ..., *acr*(*k*), ..., *acr*(K) }

Each access control rule *acr*(*k*) is comprised of three type of components, denoted accessControlOriginators, accessControlOperations and accessControlContexts. The accessControlContext component is an optional parameter.

Hence, an access control rule *acr*(*k*) is either represented as a pair:

*acr*(*k*) = {*acr*(*k*)\_accessControlOriginators, *acr*(*k*)\_accessControlOperations}

or as a 3-tuple:

*acr*(*k*) = {*acr*(*k*)\_accessControlOriginators, *acr*(*k*)\_accessControlOperations, *acr*(*k*)\_accessControlContexts}

The generic term "access-control-rule-tuple" is used when referring to a rule *acr*(*k*).

A set *acrs* of access control rules may consist of a mix of pairs and 3-tuples. For pairs, any context parameters associated with a request message are admissible.

The three component parameters of an access-control-rule-tuple supported in the present document are shown in table 7.1.3-1.

Table 7.1.3-1: Parameters of an access-control-rule-tuple

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Usage Description | Mandatory/Optional | Format |
| accessControlOriginators | Set of Originators that can be authorized | M | List of CSE-IDs and/or AE‑IDs, or keyword "all" to grant access to all originators |
| accessControlOperations | Set of Operations that can be authorized | M | Enumerated list of operations Create Retrieve, Update, Delete, Discover, Notify |
| accessControlContexts | See table 7.1.3-3 | O | See table 7.1.3-3 |
| accessControlObjectDetails | See table 7.1.3-2 | O | See table 7.1.3-2 |
| accessControlAuthenticationFlag | Indicates whether the rule applies only to Originators which are considered to be authenticated by the Hosting CSE | O | Boolean |
| accessControlAttributes | Set of resource attributes for which access can be authorized | O | List of resource attribute name(s). |

The accessControlOriginators parameter comprises a list of SP domain names, CSE-IDs, AE-IDs, resource-IDs of <group> resources and/or Role IDs of any format defined in oneM2M TS‑0001 [1]. If access for all originators is to be allowed, the reserved keyword "all" may be included into the value space of accessControlOriginators.

Using a SP domain name in accessControlOriginators means all AE-IDs and CSE-IDs matching the given domain name can be authorized.

It is furthermore allowed to use wildcard character "\*", in representations of CSE-ID and AE‑ID. The scope of a "\*" is terminated by a following "/" character. Table 7.1.3-2 shows examples of using wildcard characters in CSE-IDs and AE-IDs.

Wildcard characters are not applicable to SP domain names, resource-IDs of <group> resources and Role IDs.

Table 7.1.3-2: Examples of using wildcard characters in CSE-IDs and AE-IDs of accessControlOriginators

|  |  |  |  |
| --- | --- | --- | --- |
|  | Form of ID | Examples | Meaning |
| CSE-ID | Absolute | //m2msp.org/myCSEID  //\*/myCSEID  //\*/myCSE\* | Any CSE whose ID matches the wild cards |
|  | SP-relative | /myCSEID  /myCSE\* | Any matching CSE from the SP that is hosting the target resource |
| AE-ID | Absolute | //m2msp.org/S988  //\*/myCSEID/C9886  //\*/myCSE\*/C9886 | Any AE whose ID matches the wild cards |
|  | SP-relative | /myCSEID/C9886  /myCSEID/C98\*  /myCSE\*/C98\*  /SmyAE\* | Any matching AE from the SP that is hosting the target resource |

The data type applicable to accessControlOriginators is defined in oneM2M TS-0004 [4].

The accessControlOperations parameter comprises a list of admissible operations which can be any subset of the following elements: Create, Retrieve, Update, Delete, Discover, and Notify. While Create, Retrieve, Update, Delete, and Notify operation are explicitly indicated in the *op* parameter of a request message, the Discovery operation is indicated by ***op*** = Retrieve in combination with the provisioning of ***fc*** and *Disrestype* parameters in the request message.

The data type applicable to accessControlOperations is defined in oneM2M TS-0004 [4].

The accessControlContexts parameters are listed in table 7.1.3-3.

Table 7.1.3-3: Parameters of accessControlContexts

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Usage Description | Mandatory/Optional | Formats |
| accessControlWindow | Set of Time Windows that can be authorized | O | List of time intervals where access can be granted in extended crontab format |
| accessControlLocationRegion | Set of Location Regions that can be authorized | O | 1) Latitude/longitude coordinates, and a radius defining a circular region around the coordinates  2) Country code |
| accessControlIpAddresses | Set of IPv4 and IPv6 addresses that can be authorized | O | IPv4: dotted-decimal notation with CIDR suffix  IPv6: colon separated groups of hexadecimal digits with CIDR suffix |
| accessControlUserIDs | Set of M2M Service Users that can be authorized | O | List of M2M-User-IDs |
| accessControlEvalCriteria | Set of conditions that are factored into authorization decisions. | O | A tuple consisting of a mandatory resource identifier of a subject resource and an set of evaluation criteria applicable to the subject resource. |
| accessControlLimit | Number of times access to a resource can be authorized. | O | A number that indicates how many times access can be granted. |

The accessControlWindow parameter represents a list of elements that comply with the extended crontab syntax as defined in clause 7.3.8 of oneM2M TS-0004 [4]. It allows definition of periodically recurring time intervals at which access can be granted, when the ***rq\_time*** parameter associated with the access request message falls into such interval.

For the elements of accessControlLocationRegion there are two representation choices. These can be represented by a 2‑character country code or a circle with radius *R* centred at a point defined in terms of longitude and latitude parameters. Refer to Annex F for detailed information. Each element of accessControlLocationRegion defines an admissible location region, which is compared with the ***rq\_loc*** parameter associated with the access request message.

The data types applicable to accessControlLocationRegion and ***rq\_loc*** are defined in oneM2M TS-0004 [4].

The accessControlIpAddresses parameter represents a list of IPv4 and IPv6 addresses in dotted-decimal notation with CIDR suffix or colon separated groups of hexadecimal digits with CIDR suffix, respectively. If the ***rq\_loc*** parameter associated with the access request message matches one of these addresses, access may be granted with regard to this criterion.

The data types applicable to accessControlIpAddresses and ***rq\_ip*** are defined in oneM2M TS-0004 [4].

The accessControlUserIDs parameter comprises a list of M2M-User-IDs having a format defined in oneM2M TS‑0001 [1].

This accessControlEvalCriteria parameter represents the conditions determining if the request operation is to be allowed. It allows conditional access to the resource based on conditions not contained in the received request. The accessControlEvalCriteria parameter is a tuple that consists of a mandatory subjectResourceID element as defined in table 9.6.61-2 in oneM2M TS‑0001 [1] and an evalCriteria element defined in table 9.6.61-3 in oneM2M TS-0001 [1].

The accessControlLimit parameter represents the number of times that the policy defined by an access-control-rule-tuple can allow authorization to the requested resource. This attribute maintains the number of remaining accesses allowed. The parameter is decremented each time an access to the requested resource is granted. If the value is greater than zero then access is granted, otherwise access is denied. If the accessControlLimit parameter is not present in an access-control-rule-tuple, then there are no restrictions on the number of times access is granted.

Using just a SP domain name in accessControlUserIDs means all M2M-User-IDs matching the given domain name can be authorized. For example, “//m2msp.org”.

It is furthermore allowed to use a wildcard character "\*" within the SP-Relative-M2M-User-ID portion of a M2M-User-ID. For example, //m2msp.org/homeowner\*. A wildcard character is not permitted within the SP domain name portion of a M2M-User-ID.

The data type applicable to accessControlUserIDs is defined in oneM2M TS-0004 [4].

The accessControlAuthenticationFlag parameter is a Boolean value. If the accessControlAuthenticationFlag parameter is not present, then the value is assumed to be FALSE. If the accessControlAuthenticationFlag parameter is TRUE, then this indicates that the access control rule applies only to Originators considered to have been authenticated by the Hosting CSE. Clause 7.1.2 specifies the criteria used to decide whether or not the Originator is considered to have been authenticated by the Hosting CSE.

The accessControlObjectDetails parameters are listed in table 7.1.3-4.

Table 7.1.3-4: Parameters of accessControlObjectDetails

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Usage Description | Mandatory/Optional | Formats |
| resourceType | Resource type on which access control rule applies | O | Resource type identifier |
| specializationType | Identifier of mgmtDefinition or containerDefinition | O | mgmtDefinition or containerDefinition represented as a string. |
| childResourceType | Set of resource type identifiers that can be created under the parent resource. | O | Resource type list. |

The accessControlObjectDetails attribute specifies a subset of child resource types of the targeted resource to which the access control rule applies. If an access control rule includes *accessControlObjectDetails*, then *childResourceType* is specified. An access control rule which does not include any *accessControlObjectDetails* parameters applies to all child resource types of the target resource. The *accessControlObjectDetails* parameter is described in table 9.6.2.4-1 of oneM2M TS‑0001 [1]. Child resource types listed in the *childResourceType* component are subject of access control for the Create operation only. Once a child resource is created, the Access Control Policies assigned directly to it apply. The *resourceType* and *specializationType* elements are optional. If either the *resourceType* or *specializationType* element is present in *accessControlObjectDetails*, the CSE matches the type of resource or specialization of the targeted resource with the value specified in the *resourceType* or *specializationType* element. Further checking of *childResourceType* is done only if the *resourceType* or *specializationType* match occurs. However, if the *resourceType* and *specializationType* elements are not provided, then only *childResourceType* match is performed.

### 7.1.4 Access Control Decision

The access decision is derived by comparing the parameters associated with a resource access request message as described in clause 7.1.2 with the access control rules included in the *privileges* or *selfPrivileges* attributes of all ACP sets assigned to the protected resource by means of the *accessControlPolicyIDs*, see figure 7.1.1-1.

The result of the access decision algorithm, i.e. the access decision, is the overall result of evaluating the applicable set of access control rules, *acrs*, against the parameters associated with the access request message. This access decision can be represented by a value of binary data type. The overall result of the access decision algorithm is denoted here with the variable name *res\_acrs*:



The reference access decision algorithm is specified in clause 7.1.5. For any given sets of inputs, an implementation of the access decision processing shall return the same result as the reference access decision algorithm would return for those inputs.

If the access decision algorithm yields the result *res\_acrs* = TRUE, then the access decision for the requested resource shall be "Permit".

If the access decision algorithm yields the result *res\_acrs* = FALSE, or the access decision algorithm is not capable of deriving a final result (e.g. due to indeterminate parameters), then the access decision for the requested resource shall be "Deny".

### 7.1.5 Description of the Access Decision Algorithm

The reference access decision algorithm specified in this clause combines partial access control results obtained for each of the individual access control rules contained in a *privileges* or *selfPrivileges* attribute. Further, if multiple ACP instances are assigned to the protected resource, the reference access decision algorithm combines the partial access control results obtained for the individual ACPs of an ACP set.

The algorithm specified in this clause adopts a "Permit-overrides" combining algorithm with respect to access control rules and ACPs as defined in XACML [i.5]. This algorithm has the following behaviour:

1. If a decision is "Permit" for only a single access control rule included in the *privileges* (or *selfPrivileges*) attribute of a single ACP, the result is "Permit".
2. Otherwise, the result is "Deny".

The logic for evaluating a request against a privilege can be described mathematically as follows. A *privileges* or *selfPrivileges* attribute included in an <*accessControlPolicy*> resource represents a set of access control rules, *acrs*, which is built as in figure 7.1.5-1.



Figure 7.1.5-1: Logic to evaluate privileges in the reference access decision algorithm

The parameters associated with a request, which are evaluated against the parameters contained in the access control rules are specified in clause 7.1.3.

The access decision *res\_acrs* defined in clause 7.1.4 is derived by evaluating whether or not the parameters associated with the request message listed in tables 7.1.2-1 and 7.1.2-2 match any of the access control rules contained in the access control rule set defined in clause 7.1.3 as follows:

*res\_acrs* = *res\_acr*(1) OR *res\_acr*(2) ... OR *res\_acr*(k) … OR *res\_acr*(K),

where *res\_acr*(*k*) represents the logical evaluation result (i.e. TRUE/FALSE or 1/0) of the request parameters against the *k*th access control rule in the set *acrs*, which can be expressed as follows:

*res\_acr*(*k*) = *res\_authn(k)* AND *res\_origs*(*k*) AND *res\_ops*(*k*) AND *res\_ctxts*(*k*) AND *res\_objd*(*k*) AND *res\_attrs*(*k*),

where *k* = 1…K.

The first partial logical result variable *res\_authn(k)* on the right side of above equation shall be evaluated according to Table 7.1.5-1:

Table 7.1.5-1: Evaluating *res\_authn(k)*

| *acr(k)\_*accessControlAuthenticationFlag | *rq\_authn* | *res\_authn* |
| --- | --- | --- |
| TRUE | TRUE | TRUE |
| TRUE | FALSE | FALSE |
| FALSE | TRUE | TRUE |
| FALSE | FALSE | TRUE |

The remaining 5 partial logical result variables on the right side of above equation can be defined by using the following set function:



With this definition:

*res\_origs*(*k*) = ismember(***Originator***, *acr*(*k*)\_accessControlOriginators)

*res\_ops*(*k*) = ismember(***Operation***, acr(*k*)\_ accessControlOperations)

In the above equation, the ***Originator*** variable refers to the authenticated identity of the originator of the request primitive which matches the ***From*** parameter.

The third partial logical result *res\_ctxts*(*k*) is derived as follows:

*res\_ctxts*(*k*) = *res\_context*(*k*, 1) ... OR *res\_context*(*k*, *m*) ... OR *res\_context*(*k,* M\_*k*),

where:

*res\_context*(*k*, *m*) = *res\_time*(*k*, *m*) AND *res\_ip*(*k*, *m*) AND *res\_loc* (*k*, *m*) AND *res\_uids(k, m)*, k = 1…K, *m* = 1…M\_*k*

and

*res\_time*(*k*, *m*) = ismember(***rq\_time***, *acr*(*k*)\_accessControlWindow(*m*))

*res\_ip*(*k*, *m*) = ismember(***rq\_ip***, *acr*(*k*)\_accessControlIpAddresses(*m*))

*res\_loc* (*k*, *m*) = ismember(***rq\_loc***, *acr*(*k*)\_accessControlLocationRegion(*m*))

*res\_uids*(*k*, *m*) = ismember(***M2M Service User***, *acr*(*k*)\_accessControlUserIDs(*m*))

The fourth partial logical result *res\_objd*(*k*) applies to Create request primitives only and is derived as

*res\_ objd*(*k*) = *res\_ objdetails*(*k*, 1) ... OR *res\_ objdetails*(*k*, *m*) ... OR *res\_ objdetails*(*k,* M\_*k*),

where:

*res\_ objdetails*(*k, m*) = *res\_resourceType*(*k, m*) AND *res\_specializationType*(*k, m*) AND *res\_childResource*(*k,m*),

for *m* = 1…M\_*k***.** The three logical arguments are defined below.

For each given element *acr*(*k*)\_accessControlObjectDetails(*m*) in an access control rule determine if the optional *resourceType* parameter is present

*resourceType* = *acr*(*k*)\_accessControlObjectDetails(*m*)/resourceType

Depending on the presence of *resourceType*, *res\_resourceType*(*k, m*) is derived as



where *targetResourceTypeID* is the resource type identifier associated with the resource addressed in the ***To*** parameter of the Create request primitive.

If the value of the *resourceType* element is 13 (<mgmtObject> specialization) or 28 (<flexContainer> specialization>), the optional specializationType element shall also be included in accessControlObjectDetails:

*specializationType* = *acr*(*k*)\_accessControlObjectDetails(*m*)/specializationType

If *specializationType* is present, it shall be matched against the *mgmtDefinition* or *containerDefinition* attributes given in the ***Content*** parameter of the Create request primitive.



The *childResourceType* element is mandatory in any given accessControlObjectDetails element of an access control rule. It includes a list of *j* = 1…J child resource type identifiers to which the rule applies. The jth list element is denoted as follows

*childResourceType*(*k*, *m*. *j*) = *acr*(*k*)\_accessControlObjectDetails(*m*)/childResourceType(*j*), *j* = 1…J

The logical variable *res\_childResource*(*k, m*) is derived as

*res\_ childResource* (*k, m*) = ismember(***Resource Type***, *childResourceType*(*k*, *m*, *j*))

where ***Resource Type*** refers to the value of the parameter of the given Create request primitive.

NOTE: If resourceType and specializationType are not present in acr(k)\_accessControlObjectDetails(m), res\_ objdetails(k, m) = res\_resourceType(k, m) AND res\_specializationType(k, m) AND res\_childResource(k,m) = res\_childResource(k,m).

The fifth partial logical result *res\_attrs*(*k*) is derived as follows:

*res\_attrs*(*k*) = ismember(***rq\_attributes****, acr*(*k*)*\_*accessControlAttributes),

where ***rq\_attributes*** refers to the targeted attributes specified in the ***To*** or ***Content*** parameter of the request.

If all the requested attribute names referenced by ***rq\_attributes*** match the names of attributes present in  *acr*(*k*)*\_accessControlAttributes*, then *res\_attrs*(*k*) is True or 1, otherwise *res\_attrs*(*k*) is False or 0.

NOTE: Attribute level access control checks are only performed for access-control-rule-tuples that include an accessControlAttributes parameter. If an access-control-rule-tuple does not include an *accessControlAttributes* parameter, then only resource level access control checks are performed for that rule.

Thanks to the "Permit-overrides" combining approach, if the access control decision for one access control rule results in *res\_acr* = TRUE, the reference access decision algorithm can stop without evaluating any other applicable access control rules of the current ACP or any other ACPs in the ACP set, and the final access decision is "Permit".

### \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of Change 1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*