|  |
| --- |
|  |

|  |  |
| --- | --- |
| CHANGE REQUEST | |
| Meeting ID:\* | SDS 42 |
| Source:\* | Bob Flynn, Convida Wireless , Bob.Flynn@convidawireless.com |
| Date:\* | 2019-09-11 |
| Reason for Change/s:\* | Non-Originator context ACPS |
| CR against: Release\* | Rel-4 |
| CR against: WI\* | Active < WI-0077>  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-0001 v4.1.0 |
| Clauses \* | 5.4.3 |
| 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 |
| Other TS/TR(s) impacted | None |
| 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 2019 (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

TR-0050 section 6.3.4 presents a solution for Non-Originator Context.

This contribution incorporates that solution into TS-0001.

Note: there is a tangential change that I discovered while preparing this contribution that I fix in change 1.

*eventCriteria* was used in the description of <action> in early drafts that were changed to *evalCriteria*. Related typo is fixed.

Change 2: Add the *evalCriteria* parameter to the *accessControlContexts* attribute of an <accessControlPolicy>. In a manner similar to other “contexts”, when present the *evalCriteria* SHALL evaluate to **true** for the requested operation to be allowed.

TR-0050 section 6.3.5 presents a solution for ACPs with limited usage configurations.

Change 2: add *accessControlLimit* parameter to the *accessControlContexts* attribute of an <accessControlPolicy>. In a manner similar to other “contexts”, when present the *accessControlLimit* value SHALL be greater than zero for the requested operation to be allowed.

TR-0050 section 6.3.6 presents a solution for improved default behavior for ACPs when resources are created.

Change 3: Adds new parameters to *accessControlObjectDetails* attribute of an <accessControlPolicy>

TR-0050 section 6.3.7 presents a solution for identifying the permissions that an originator has on a target resource.

Change 4: defines a new **Result Content** type **Permission** and it defines a new ***filter Criteria*** type *operations.*

The rcn= Permission returns the permissions that the originator has for the target resource.

The new *filter Criteria operations* supports specifying the specific operations that the originator wants to be present for the resources returned by the request. For example a discovery request where the filter criteria specifies labels=parkingLot and operations = UPDATE

R02 –

During review we were not able to achieve consensus on the manner that ACP propagation is defined. We agreed that a solution should be explored. We agreed to remove that content from this contribution and explore it further in a separate contribution.

Therefore change 3 is removed

R01 –

For the non-originator contexts, clarify that the evaluation may not be related to the originator of the request.

For ACP Propagation, ensure that this does not create a ACP vulnerability by stating that the level of propagation applies to levels relative to the location of the parent resource of the ACP resource and that duplicated ACPs do not have these propagation parameters set.

For example:

<AE1>

<ACP1> - propagation level set to 1 means that this will apply to children of <AE1>

<ACP2> - propagation level set to 2 means that this will apply to grand-children of <AE1>

<container1> - <ACP1> and <ACP2> are applied to *acpids* (only if created with no *acpids*)

<container2> - <ACP2> is applied to *acpids* (only if created with no *acpids*)

<container3> - nothing added to *acpids*

R02 – Better example showing multiple <AEn> how does provisioning get done or avoided.

Consider a new resource type,

-------------------------------------------------- Start of Change 1--------------------------------------------------

### Resource Type *action*

The <*action*> resource enables the system to specify which actions or operations to be performed on system resources (e.g. devices or services) based on monitored events. The events may include state changes of subject resources, requests from applications, etc. Once the event occurs, and if specified <dependency> conditions are met, the Hosting CSE sends a primitive defined by <action> resource attributes.

These child resources and attributes provide information about:

* The subject resource, which is the resource monitored to determine if a primary event occurs. Action is to be triggered if the primary event is the change of the state of the subject resource .
* The object resource, which is the resource which is the target of the triggered action
* The input resource, which is a resource which may provide input parameters for the action. Alternatively, an input value may be provided.
* Conditions to be monitored in order to determine if the primary event occurs
* Dependencies to be evaluated in order to determine if the action is to be conditionally performed.
* Parameters providing priorities of the action and priorities of the dependencies.

The *<action>* resource shall contain the child resources specified in table 9.6.61-1.

Table 9.6.61-1: Child resources of *<action>* resource

| Child Resources of *<action>* | Child Resource Type | Multiplicity | Description | *<actionAnnc>* Child Resource Types |
| --- | --- | --- | --- | --- |
| *[variable]* | *<dependency>* | 0..n | See clause 9.6.62 | *<dependencyAnnc>* |
| *[variable]* | *<subscription>* | 0..n | See clause 9.6.8 | *<subscription>* |

The *<action>* resource shall contain the attributes specified in table 9.6.61-2.

Table 9.6.61-2: Attributes of *<action>* resource

| Attributes of  *<action>* | Multiplicity | RW/  RO/  WO | Description | *<actionAnnc>* Attributes |
| --- | --- | --- | --- | --- |
| *resourceType* | 1 | RO | See clause 9.6.1.3. | NA |
| *resourceID* | 1 | RO | See clause 9.6.1.3. | NA |
| *resourceName* | 1 | WO | See clause 9.6.1.3. | NA |
| *parentID* | 1 | RO | See clause 9.6.1.3. | NA |
| *expirationTime* | 1 | RW | See clause 9.6.1.3. | MA |
| *accessControlPolicyIDs* | 0..1 (L) | RW | See clause 9.6.1.3. | MA |
| *Labels* | 0..1 (L) | RW | See clause 9.6.1.3. | MA |
| *creationTime* | 1 | RO | See clause 9.6.1.3. | NA |
| *lastModifiedTime* | 1 | RO | See clause 9.6.1.3. | NA |
| *stateTag* | 1 | RO | See clause 9.6.1.3. | OA |
| *announceTo* | 0..1 (L) | RW | See clause 9.6.1.3. | NA |
| *announcedAttribute* | 0..1 (L) | RW | See clause 9.6.1.3. | NA |
| *dynamicAuthorizationConsultationIDs* | 0..1 (L) | RW | See clause 9.6.1.3. | OA |
| *Creator* | 1 | RO | See clause 9.6.1.3. | NA |
| *actionSubjectResource* | 0..1 | RW | The resource that is the subject to be monitored for triggering the evaluation of the *evalCriteria* attribute. The action indicated by the *operation* attribute Action is to be triggered based on the change of the state of the resource indicated the *actionSubject Resource* attribute.The subject can be any resource in the system. If the attribute is missing the subject is the parent resource. | OA |
| *actionPriority* | 0..1 | RW | Attribute indicating the priority of the action when compared with the other actions with the same event expression (*subjectResourceID* and *evalCriteria*), with highest priority indicated by the lowest value. Local policies are applied for equal or unspecified priorities. | OA |
| *subjectResourceID* | 0..1 | RW | The *resourceID* of the resource that is the subject of monitoring for evaluation of the *evalCriteria* attribute. The subject can be any resource in the system. If the attribute is missing the subject is the parent resource of this <action> resource. | OA |
| *evalCriteria* | 0..1 | RW | This attribute provides the conditions determining if the action is to be conditionally triggered to the object resource.  The *evalCriteria* attribute, in conjunction with the *subjectResourceID,* forms the event expression that is being monitored. See further description below and in table 9.6.61-3. | OA |
| *evalMode* | 1 | RW | Attribute provides the macro control mode of the evaluation of the <action> resource. Some values are:  0 = off  1 = once  2 = periodic  3 = continuous | OA |
| *evalControlParam* | 0..1 | RW | When *evalMode*= periodic, *evalControlParam* represents the periodicity.  When *evalMode*= continuous the evaluation is restarted as soon as an action has been triggered. In this case, if *evalControlParam* is specified, it determines the number of times the Hosting CSE shall trigger the event , otherwise it is repeated indefinitely.  When *evalMode* is off or once this attribute can be ignored. | OA |
| *evalPriority* | 0..1 (L) | RW | List of priorities to be applied for the evaluation of the dependencies . This is an ordered list of the *resourceIDs* of the child <dependency> resources. | OA |
| *objectResourceID* | 0..1 | RW | The *resourceID* of the target of the primitive to be sent when the event specified by *evalCriteria* occurs at the subject resource. The target may be specified also as the URI of a resource attribute. | MA |
| *actionPrimitive* | 1 | RW | This attribute stores the entire content of the primitive associated with this action with its parameters (e.g. the op parameter indicating the operation to be performed on the resource identified by objectResourceID), with some parameters being overwritten by the Hosting CSE when the action is triggered as follows:   * the *to* and *resource type* parameters will be derived from *objectResourceID* attribute. * the *content* parameter is derived from the *input* attribute.   When *objectResourceID* or *content* the *input* attributes are empty, the respective parameters are maintained as indicated by *actionPrimitive.* | MA |
| *input* | 0..1 (L) | RW | Input parameters for the operation, used by the Hosting CSE to overwrite the *content* parameter of the *actionPrimitive* before sending the primitive, at the time that the action is triggered. The attribute could be one of the following:   * a string * a resourceID * the URI of a resource attribute.   The Hosting CSE overwrites the c*ontent* parameter of *actionPrimitive* with the following (respectively):   * the given string * the resource representation of the resource with the given *resourceID.* * the value of the given resource attribute | OA |
| *outputResourceID* | 0..1 | RW | The *resourceID* of a resource where the results of the action are stored. | OA |
| *actionResult* | 1 | RW | The result of the action. If *outputResourceID* is set, the Hosting CSE performs a RETRIEVE based on it when the response primitive of the action primitive is received and stores its content. Otherwise, the response primitive of the action primitive is stored in this attribute. | OA |

The conditions represented in the *evalCriteria* attribute determine if the action is to be conditionally triggered.

The *evalCriteria* attribute is comprised of triples (*subject*, *operator*, *threshold*) with the parameters shown and described in table 9.6.61-3.

Table 9.6.61-3: Parameters in *evalCriteria* triple

| Name | Description |
| --- | --- |
| *subject* | This parameter shall indicate an attribute of the *subjectResourceID* resource e.g. “tempContainer.content” |
| *operator* | This parameter is a keyword used to construct the evaluation logic, e.g. ‘equals’, ‘not equals’, ‘greater than’, ‘less than’, ‘greater or equal’, ‘less or equal’, ‘string match’, etc. |
| *threshold* | This parameter provides a value used to evaluate the criteria. The value type shall match the type of the attribute used as *subject*. |

-------------------------------------------------- End of Change 1---------------------------------------------------

-------------------------------------------------- Start of Change 2--------------------------------------------------

#### 9.6.2.2 *accessControlContexts*

The *accessControlContexts* is an optional parameter in an access-control-rule-tuple that contains a list, where each element of the list, when present, represents a context that is permitted to use this access control rule. Each request context is described by a set of parameters, where the types of the parameters can vary within the set. Table 9.6.2.2-1 describes the supported types of parameters in *accessControlContexts*.

The following Originator *accessControlContexts* shall be considered for access control policy check by the CSE.

Table 9.6.2.2-1: Types of Parameters in *accessControlContexts*

| Name | Description |
| --- | --- |
| *accessControlTimeWindow* | Represents a time window constraint which is compared against the time that the request is received at the Hosting CSE. |
| *accessControlLocationRegion* | Represents a location region constraint which is compared against the location of the Originator of the request. |
| *accessControlIpAddress* | Represents an IP address constraint or IP address block constraint which is compared against the IP address of the Originator of the request. |
| *accessControlEvalCriteria* | This attribute provides 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 consists of a mandatory *subjectResourceID* attribute as defined in table 9.6.61-2 and the *evalCriteria* attribute described in table 9.6.61-3. NOTE: this uses the same definitions that are present in the <action> resource, but does not use the <action> resource. |
| *accessControlLimit* | Represents the number of times that the policy defined in this accessControlRule can allow authorization to the requested resource. This attribute maintains of the number of authorizations granted based on this policy. This value is decremented each time the evaluation grants access to the requested resource. If this value is greater than zero (0) then the request operation is allowed. If the *accessControlLimit* parameter is not present then the request operation is allowed (unlimited access). |

-------------------------------------------------- End of Change 2---------------------------------------------------



-------------------------------------------------- Start of Change 4--------------------------------------------------

### 8.1.2 Request

Requests over the Mca and Mcc reference points, from an Originator to a Receiver, shall contain mandatory and may contain optional parameters. Certain parameters may be mandatory or optional depending upon the Requested operation. In this clause, the mandatory parameters are detailed first, followed by those that are operation dependent, and then by those that are optional:

* ***To*:** Address of the target resource or target attribute for the operation. The ***To*** parameter shall conform to clause 9.3.1.

NOTE 1: ***To*** parameter can be known either by pre-provisioning (clause 11.2) or by discovery (clause 10.2.6 for discovery). Discovery of *<CSEBase>* resource is not supported in this release of the document. It is assumed knowledge of *<CSEBase>* resource is by pre-provisioning only.

NOTE 2: The term target resource refers to the resource which is addressed for the specific operation. For example, the ***To*** parameter of a Create operation for a resource *<example>* would be "/m2m.provider.com/exampleBase". The ***To*** parameter for the Retrieve operation of the same resource *<example>* is "/m2m.provider.com/exampleBase/example".

NOTE 3: For Retrieve operation (clause 10.1.3), the ***To*** parameter can be the URI of an attribute to be retrieved.

* ***From*:** Identifier representing the Originator.

The ***From*** parameter is used by the Receiver to check the Originator identity for access privilege verification.

* ***Operation*:** operation to be executed: Create (C), Retrieve (R), Update (U), Delete (D), Notify (N).

The ***Operation*** parameter shall indicate the operation to be executed at the Receiver:

* **Create (C): *To*** is the address of the target resource where the new resource (parent resource).
* **Retrieve (R):** an existing ***To*** addressable resource is read and provided back to the Originator.
* **Update (U):** the content of an existing ***To*** addressable resource is replaced with the new content as in ***Content*** parameter. If some attributes in the ***Content*** parameter do not exist at the target resource, such attributes are created with the assigned values. If some attributes in the ***Content*** parameter are set to NULL, such attributes are deleted from the addressed resource.
* **Delete (D):** an existing ***To*** addressable resource and all its sub-resources are deleted from the Resource storage.
* **Notify (N):** information to be sent to the Receiver, processing on the Receiver is not indicated by the Originator.
* ***Request Identifier*:** request Identifier (see clause 7.1.7).

Example usage of request identifier includes enabling the correlation between a Request and one of the many received Responses.

**Operation dependent Parameters:**

* ***Content*:** resource content to be transferred.

The ***Content*** parameter shall be present in Request for the following operations:

* **Create (C):** ***Content*** is the content of the new resource with the resource type ***ResourceType.***
* **Update (U):** ***Content*** is the content to be replaced in an existing resource. For attributes to be updated at the resource, ***Content*** includes the names of such attributes with their new values. For attributes to be created at the resource, ***Content*** includes names of such attributes with their associated values. For attributes to be deleted at the resource, ***Content*** includes the names of such attributes with their value set to NULL.
* **Notify (N):** ***Content*** is the notification information.

The ***Content*** parameter may be present in Request for the following operations:

* **Retrieve (R):** ***Content*** is the list of attribute names from the resource that needs to be retrieved. The values associated with the attribute names shall be returned.
* ***Resource Type:*** type of resource.

The ***ResourceType*** parameter shall be present in Request for the following operations:

* **Create (C):** ***Resource Type*** is the type of the resource to be created.

**Optional Parameters:**

* ***Role IDs:*** optional, required when role based access control is applied. A list of Role-IDs that are allowed by the service subscription shall be provided otherwise the request is considered not valid.

The ***Role*** ***IDs*** parameter shall be used by the Receiver to check the Access Control privileges of the Originator.

* ***Originating Timestamp*:** optional originating timestamp of when the message was built.

Example usage of the originating timestamp includes: to measure and enable operation (e.g. message logging, correlation, message prioritization/scheduling, accept performance requests, charging, etc.) and to measure performance (distribution and processing latency, closed loop latency, SLAs, analytics, etc.)

* ***Request Expiration Timestamp*:** optional request message expiration timestamp. The Receiver CSE should handle the request before the time expires. If a Receiver CSE receives a request with ***Request Expiration Timestamp*** with the value indicating a time in the past, then the request shall be rejected.

Example usage of the request expiration timestamp is to indicate when request messages (including delay‑tolerant) should expire and to inform message scheduling/prioritization. When a request with set expiration timestamp demands an operation on a Hosting CSE different than the current Receiver CSE, then the current CSE shall keep trying to deliver the Request to the Hosting CSE until the request expiration timestamp time, in line with provisioned policies.

* ***Result Expiration Timestamp*:** optional result message expiration timestamp. The Receiver CSE should return the result of the request before the time expires.

Example usage of the result expiration timestamp: An Originator indicates when result messages (including delay-tolerant) should expire and informs message scheduling/prioritization. It can be used to set the maximum allowed total request/result message sequence round trip deadline.

* ***Response Type*:** optional response message type: Indicates what type of response shall be sent to the issued request and when the response shall be sent to the Originator:
* **nonBlockingRequestSynch**: In case the request is accepted by the Receiver CSE, the Receiver CSE responds, after acceptance, with an Acknowledgement confirming that the Receiver CSE will further process the request. The Receiver CSE includes in the response to an accepted request a reference that can be used to access the status of the request and the result of the requested operation at a later time. Processing of Non-Blocking Requests is defined in clause 8.2.2 and in particular for the synchronous case in clause 8.2.2.2.
* **nonBlockingRequestAsynch {optional list of notification targets}:** In case the request is accepted by the Receiver CSE, the Receiver CSE shall respond, after acceptance, with an Acknowledgement confirming that the Receiver CSE will further process the request. The result of the requested operation needs to be sent as notification(s) to the notification target(s) provided optionally within this parameter as a list of entities or to the Originator when no notification target list is provided. When an empty notification target list is provided by the Originator, no notification with the result of the requested operation shall be sent at all. Processing of Non‑Blocking Requests is defined in clause 8.2.2 and in particular for the asynchronous case in clause 8.2.2.3.
* **blockingRequest:** In case the request is accepted by the Receiver CSE, the Receiver CSE responds with the result of the requested operation after completion of the requested operation. Processing of Blocking Requests is defined in clause 8.2.1. This is the default behaviour when the *Response Type* parameter is not given the request.
* **flexBlocking {optional list of notification targets}:** When ***Response Type*** in the request received by the Receiver CSE is set to flexBlocking, it means that the Originator of the request has the capability to accept the following types of responses: nonBlockingRequestSynch, nonBlockingRequestAsynch and blockingRequest.

The Receiver CSE shall make the decision to respond using blocking or non-blocking based on its own local context (memory, processing capability, etc.) if not defined in the resource handling procedure.

If the Receiver CSE choose to respond using non-blocking mode or blocking mode, based on the presence of notification targets in the request:

* If the notification targets are provided in the request and the Receiver CSE is responding, the Receiver CSE shall choose and respond with nonBlockingRequestAsynch, nonBlockingRequestSynch or blockingRequest mode.
* If notification targets are not provided, the Receiver CSE shall choose and respond with nonBlockingRequestSynch or blockingRequest mode.
  + **No Response:**In case the request is accepted by the Receiver CSE or AE, the Receiver CSE or AE does not respond with the result of the requested operation after completion of the requested operation. Note, in this case the Result Content parameter should not be included in the request.

Example usage of the response type set to *nonBlockingRequestSynch*: An Originator that is optimized to minimize communication time and energy consumption wants to express a Request to the receiver CSE and get an acknowledgement on whether the Request got accepted. After that the Originator may switch into a less power consuming mode and retrieve a Result of the requested Operation at a later time.

Further example usage of response type set to *nonBlockingRequestSynch:* When the result content is extremely large, or when the result consists of multiple content parts from a target group which are to be aggregated asynchronously over time.

* ***Result Content*:** optional result content: Indicates what are the expected components of the result of the requested operation. This shall be indicated in the ***Result Content*** parameter. Settings of ***Result Content*** depends on the requested operation specified in ***Operation***. This parameter is not applicable when ***Response Type*** has a value of *No Response*. Possible values of ***Result Content*** are:
* **attributes:** A representation of the targeted resource including all its attributes shall be returned as content, without the address(es) of the child resource(s) or their descendants. For example, if the request is to retrieve a *<container>* resource, the address(es) of the *<contentInstance>* child-resource(s) is not provided. This setting shall be only valid for Create, Retrieve, Update, or Delete operation. If the Originator does not set ***Result Content*** parameter in a Create, Retrieve or Update request message, this setting shall be the default value when the Receiver processes the request message.
* **modified-attributes**: This setting shall be only valid for a Create or Update operation. A representation of the targeted resource including only the assigned or modified attributes relative to what was provided by the Originator of the request shall be returned as content, without the address(es) of the child resource(s) or their descendants.
* **hierarchical-address:** Representation of the address of the created resource. This setting shall only be valid for a Create operation. The address shall be in hierarchical address scheme.
* **hierarchical-address+attributes:** Representation of the address in hierarchical address scheme and the attributes of the created resource. This setting shall only be valid for a Create operation.

- **attributes+child-resources:** Representation of the requested resource, along with a nested representation of all of its child resource(s), and their descendants, in line with any provided filter criteria as given in the ***Filter Criteria*** parameter shall be returned as content. If there is no filter criteria parameter in the request message, then all children/descendants are returned along with their attributes. For example, if the request is to retrieve a *<container>* resource that only has *<contentInstance>* children, the attributes of that *<container>* resource and a representation of all of its *<contentInstance>* child-resource(s), including their attributes, are provided.

The originator may request to limit the maximum number of allowed nesting levels. The originator may also include an offset that indicates the starting point of the direct child resource. The offset shall start at 1. The hosting CSE shall return all direct child resources and their descendants, or up to the maximum nesting level specified in a request subject to maximum size limit that may be imposed by the hosting CSE. The offset, maximum number/size and maximum level shall be specified in ***Filter Criteria*** as *offset*, *limit*, and *level* condition, respectively, by the Originator.

The hosting CSE shall list parent resources before their children. This means that the originator of the request will not receive a discovered resource without having received its parents. The hosting CSE shall also ensure that proper nesting representation of all the children is incorporated in its listing for parents and children.

Nested processing is applicable at every level in the resource tree. If a direct child resource and all its descendants cannot be included in the returned content due to size limitations imposed by the hosting CSE then the direct child resource shall not be included in the response.

An indication shall be included in the response signalling if the returned content is partial. If the indication is for partial content, the response shall include an offset for the direct child resource where processing can restart for the remaining direct child resources

This shall be only valid for a Retrieve/Delete operation.

* **child-resources:** A nested representation of the resource's child resource(s) their descendants and their attributes shall be returned as content. The resources that are returned are subject to any filter criteria that are given in the ***Filter Criteria*** parameter (if there are no filter criteria then all children and their descendants are returned). The attributes of the parent resource are not returned, but all the attributes of the children are returned. For example, if the request is to retrieve a *<container>* resource that only has *<contentInstance>* children, only a representation of all of its *<contentInstance>* child-resource(s) is provided.

The offset, maximum number/size and maximum level shall be specified in ***Filter Criteria*** as *offset*, *limit*, and *level* condition, respectively, by the Originator. Processing of direct child resources, size limitations, maximum nesting level, and offset for the starting of direct child resource processing of **the attributes+child-resources** option shall apply to this option as well.

This shall be only valid for a Retrieve/Delete operation.

* **attributes+child-resource-references:** Representation of the requested resource, along with the address(es) of the child resource(s), and their descendants shall be returned as content. For example, if the request is to retrieve a *<container>* resource, the *<container>* resource and the address(es) of the *<contentInstance>* child-resource(s) are provided.

The offset, maximum number/size and maximum level shall be specified in ***Filter Criteria*** as *offset*, *limit*, and *level* condition, respectively, by the Originator. Processing of child resources, size limitations, maximum nesting level, and offset for the starting of child resource processing of **the attributes+child-resources** option shall apply to this option as well.

This shall be only valid for a Retrieve/Delete operation.

* **child-resource-references:** Address(es) of the child resources and their descendants, without any representation of the actual requested resource shall be returned as content. For example, if the request is to retrieve a *<container>* resource, only the address(es) of the *<contentInstance>* child-resource(s) is provided.

The offset, maximum number/size and maximum level shall be specified in ***Filter Criteria*** as *offset*, *limit*, and *level* condition, respectively, by the Originator. Processing of child resources, size limitations, maximum nesting level, and offset for the starting of child resource processing of **the attributes+child-resources** option shall apply to this option as well.

This shall be only valid for a Retrieve/Delete operation.

This option can be used within the context of resource discovery mechanisms (see clause 10.2.6).

* **nothing:** Nothing shall be returned as operational result content. If the Originator does not set the ***Result Content*** parameter in a Delete request message, this setting shall be the default value when the Receiver processes the request message. This setting shall be valid for a Create, Update, Delete, or Notify operation.

EXAMPLE: If the request is to delete a resource, this setting indicates that the response shall not include any content.

* **original-resource:** Representation of the original resource pointed by the *link* attribute in the announced resource shall be returned as content, without the address(es) of the child resource(s). This shall be only valid for a Retrieve operation where the ***To*** parameter targets the announced resource.
* **semantic-content:** Representation of semantic information that is the result of a semantic query as indicated by the setting of the ***Semantic Query Indicator*** parameter.
* **Permissions**: Representation of the permissions that the originator has for the targeted resource. The result is a consolidated representation of all the ACPs associated with the targeted resource for the originator.

Note that for any of the above options, Discovery access control is applied against discovery related procedures, while Retrieve access control procedures is applied against non-discovery related Retrieve operations.

Note that the fitter criteria usage governs the purpose of a Retrieve operation.

Table 8.1.2-1: Summary of Result Content Values

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Value** | **Create** | **Retrieve** | **Update** | **Delete** | **Notify** |
| attributes | default | default | default | valid | n/a |
| modified-attributes | valid | n/a | valid | n/a | n/a |
| hierarchical-address | valid | n/a | n/a | n/a | n/a |
| hierarchical-address+attributes | valid | n/a | n/a | n/a | n/a |
| attributes+child-resources | n/a | valid | n/a | valid | n/a |
| child-resources | n/a | valid | n/a | valid | n/a |
| attributes+child-resource-references | n/a | valid | n/a | valid | n/a |
| child-resource-references | n/a | valid | n/a | valid | n/a |
| nothing | valid | n/a | valid | default | valid |
| original-resource | n/a | valid | n/a | n/a | n/a |
| semantic-content | n/a | valid | n/a | n/a | n/a |

* ***Result Persistence*:** optional result persistence: indicates the time for which the response may persist to. The parameter is used in case of non-blocking request where the result attribute of the <request> resource should be kept at the CSE, for example, with the purpose of sharing, tracking and analytics.

In the case the response of a request is required to be kept in the CSE, for example the procedures of <request> resource, <delivery> resource and <group> resource, the ***Result Persistence*** indicates the time duration for which the CSE keeps the response available after receiving it.

Example usage of result persistence includes requesting sufficient persistence for analytics to process the response content aggregated asynchronously over time. If a result expiration time is specified, then the result persistence lasts beyond the result expiration time.

* ***Operation Execution Time*:** optional operation execution time: indicates the time when the specified operation ***Operation*** is to be executed by the target CSE. A target CSE shall execute the specified operation of a Request having its operational execution time indicator set, starting at the operational execution time. If the execution time has already passed or if the indicator is not set, then the specified operation shall be immediately executed, unless the request expiration time, if set, has been reached.

Example usage of operational execution time includes asynchronous distribution of flows, which are to be executed synchronously at the operational execution time.

NOTE 6: Time-based flows could not be supported depending upon time services available at CSEs.

* ***Event Category*:** optional event category: Indicates the event category that should be used to handle this request. Event categories are impacting how Requests to access remotely hosted resources are processed in the CMDH CSF. Selection and scheduling of connections via CMDH are driven by policies that can differentiate event categories.

Example usage of "event category" set to specific value X: When the request is demanding an operation to be executed on a Hosting CSE that is different from the current Receiver CSE, the request may be stored in the current Receiver CSE that is currently processing the request on the way to the Hosting CSE until it is allowed by provisioned policies for that event category X to use a communication link to reach the next CSE on a path to the Hosting CSE or until the request expiration timestamp is expired.

The following values for ***Event Category*** shall have a specified pre-defined meaning:

* ***Event Category* = immediate:** Requests of this category shall be sent as soon as possible and shall not be subject to any further CMDH processing, i.e. the request will not be subject to storing in CMDH buffers when communication over an underlying network is possible. In particular, CMDH processing will respect values for ***Request Expiration Timestamp***, ***Result Expiration Timestamp*** given in the original request and not fill in any default values if they are missing.
* ***Event Category* = bestEffort:** Requests of this category can be stored in CMDH buffers at the discretion of the CSE that is processing the request for an arbitrary time and shall be forwarded via Mcc on a best effort basis. The CSE does not assume any responsibility to meet any time limits for delivering the information to the next CSE. Also the maximum amount of buffered requests for this category is at the discretion of the processing CSE.
* ***Event Category***= **latest:**
* If this category is used in a request asking for a CRUD operation on a resource, the following shall apply:  
  CRUD requests using this category shall undergo normal CMDH processing as outlined further below in the present document and in oneM2M TS-0004 [3] with a maximum buffer size of one pending request for a specific pair of ***From*** and ***To*** parameters that appear in the request. If a new request message is received by the CSE with a pair of parameters ***From*** and ***To*** that has already been buffered for a pending request, the newer request will replace the buffered older request.
* If this category is used in a notification request triggered by a subscription, the following shall apply:  
  Notification requests triggered by a subscription using this category shall undergo normal CMDH processing as outlined further below in the present document and in oneM2M TS-0004 [3] with a maximum buffer size of one pending notification request per subscription reference that appears in a notification request. If a new notification request is received by the CSE with a subscription reference that has already been buffered for a pending notification request, the newer request will replace the buffered older request.
* If no further CMDH policies are provisioned for this event category, the forwarding process shall follow the 'bestEffort' rules defined above.

The M2M Service Provider shall be able to provision CMDH policies describing details for the usage of the specific Underlying Network(s) and the applicable rules as defined in the *[cmdhPolicy]* resource type for other ***Event Category*** values not listed above.

* ***Delivery Aggregation*:** optional delivery aggregation on/off: Use CRUD operations of *<delivery>* resources to express forwarding of one or more original requests to the same target CSE(s). When this parameter is not given in the request, the default behaviour is determined per the provisioned CMDH policy if available. If there is no such CMDH policy, then the default value is "aggregation off".

NOTE 7: Since ***Delivery Aggregation*** is optional, there could be a default value to be used when not present in the Request. This parameter could not be exposed to AEs via Mca.

Example usage of delivery aggregation set on: The CSE processing a request shall use aggregation of requests to the same target CSE by requesting CREATE of a *<delivery>* resource on the next CSE on the path to the target CSE.

* ***Group Request Identifier*:** optional group request identifier: Identifier optionally added to the group request that is to be fanned out to each member of the group in order to detect loops and avoid duplicated handling of operation in case of loops of group and common members between groups that have parent-child relationship.
* ***Group Request Target Members:*** optional group request target members: Indicates subset of members of a group for which fanout is to be executed. Example usage of Group Request Target Members: if fanout operation failed for some of the members then the Originator may use this parameter to execute fanout for failed members of a previous fanout operation.
* ***Filter Criteria*:** optional filter criteria: conditions for filtered operations which are described in table 8.1.2-2. This is used for resource discovery (clause 10.2.6) and general retrieve, update, delete requests (clauses 10.1.3, 10.1.4 and 10.1.5).

The Filter Criteria set includes matching conditions and filter handling conditions. Matching conditions are evaluated against resources and, when true, determine the matched resources which compose the matching result. The filter handling conditions provide additional input used to determine the filtering result (e.g. maximum number of resources to be included in the filtering result). The filtering result may be composed of one or more resources.

Example usage of retrieve requests with filter criteria using *modifiedSince* condition tag: if a target resource is modified since 12:00 then the Hosting CSE will identify it as a matched resource.

* ***Desired Identifier Result Type:*** Optional result format of resource identifiers. This parameter indicates the format of the resource identifiers in the result of operations that can return a list of resource identifiers or Child Resource References. This parameter shall take on one of the following values reflecting the options in clause 9.3.1:
* Structured identifier format.
* Unstructured identifier format.

The absence of the parameter implies that the result shall be in the form of a Structured identifier format.

* ***Token Request Indicator:*** Optional parameter used to indicate that the Originator supports the Token Request procedure, and the Originator may attempt the Token Request procedure if the Receiver provides a ***Token Request Information*** parameter in the response.
* ***Tokens:*** Optional parameter used to transport ESData-protected *Tokens* applicable to the request for use in Indirect Dynamic Authorization.
* ***Token IDs:*** Optional parameter used to transport *Token-IDs* applicable to the request for use in Indirect Dynamic Authorization.
* ***Local Token IDs:*** Optional parameter used to transport Local-Token-IDs applicable to the request for use in Indirect Dynamic Authorization.
* ***Authorization Signature Indicator***: Optional parameter used to indicate the capability for creating AuthorRelMapRecord when Originator is an AE. If the Hosting CSE does not support this parameter, the Hosting CSE should ignore it. The details of the AuthorRelMapRecord are described in clause 7.3.2.2 of oneM2M TS-0003 [2].
* ***Authorization Signature***: Optional parameter used to transport the signatures for Token(s) or TokenID(s) generated using the certificate of the AE or a MIC generated using a symmetric key shared between the AE and DAS server.
* ***Authorization Relationship Indicator***: Optional parameter used to indicate that the relationship between the AE and the Token(s) are maintained in the DAS server.
* ***Semantic Query Indicator:*** Optional parameter used to indicate whether a RETRIEVE request is a semantic query or a semantic resource discovery. If the request contains this parameter with the value set to “TRUE”, the request shall be processed as a semantic query based on the SPARQL query statement included in the “*semanticsFilter*” condition tag; other *Filter Criteria* and the following parameters shall be ignored: *Desired Identifier Result Type*, *Delivery Aggregation*. The parameter *Result Content* shall be set to **semantic-content** to indicate that the response message contains the result of a semantic query request. If it is not set or set to “FALSE” the request shall be processed as a semantic resource discovery.
* ***Release Version Indicator:*** This parameter is used to indicate the oneM2M release version that this request message conforms to. Starting with Release 2 this parameter is mandatory. The release version indicated shall apply to all oneM2M defined request parameters and certain types of content carried in the ***Content*** request parameter. Within the ***Content*** request parameter, the release version indicated shall apply to all oneM2M defined elements (e.g. notifications) and resource types with the exception of <*flexContainer*> and <*mgmtObj*> specializations which have their own version implicitly defined by their respective *containerDefinition* and *mgmtSchema* attributes. In addition, the release version indicated does not apply to resource types or specializations defined external to oneM2M.
* ***Vendor Information:*** This optional parameter is available to convey vendor specific information. The use of this parameter is not defined by oneM2M specifications.

Table 8.1.2-2: *Filter Criteria* conditions

| Condition tag | Multiplicity | Description |
| --- | --- | --- |
| **Matching Conditions** | | |
| *createdBefore* | 0..1 | The *creationTime* attribute of the matched resource is chronologically before the specified value. |
| *createdAfter* | 0..1 | The *creationTime* attribute of the matched resource is chronologically after the specified value. |
| *modifiedSince* | 0..1 | The *lastModifiedTime* attribute of the matched resource is chronologically after the specified value. |
| *unmodifiedSince* | 0..1 | The *lastModifiedTime* attribute of the matched resource is chronologically before the specified value. |
| *stateTagSmaller* | 0..1 | The *stateTag* attribute of the matched resource is smaller than the specified value. |
| *stateTagBigger* | 0..1 | The *stateTag* attribute of the matched resource is bigger than the specified value. |
| *expireBefore* | 0..1 | The *expirationTime* attribute of the matched resource is chronologically before the specified value. |
| *expireAfter* | 0..1 | The *expirationTime* attribute of the matched resource is chronologically after the specified value. |
| *labels* | 0..1 | The *labels* attribute of the matched resource matches the specified value. |
| *labelsQuery* | 0..1 | The value is an expression for the filtering of *labels* attribute of resource when it is of key-value pair format. The expression is about the relationship between label-key and label-value which may include equal to or not equal to, within or not within a specified set etc. For example, label-key equals to label value, or label-key within {label-value1, label-value2}. Details are defined in [3] |
| *childLabels* | 0..1 | A child of the matched resource has *labels* attributes matching the specified value. The evaluation is the same as for the *labels* attribute above. Details are defined in [3]. |
| *parentLabels* | 0..1 | The parent of the matched resource has *labels* attributes matching the specified value. The evaluation is the same as for the *labels* attribute above. Details are defined in [3]. |
| *resourceType* | 0..n | The *resourceType* attribute of the matched resource is the same as the specified value. It also allows differentiating between normal and announced resources. |
| *childResourceType* | 0..n | A child of the matched resource has the *resourceType* attribute the same as the specified value. |
| *parentResourceType* | 0..1 | The parent of the matched resource has the *resourceType* attribute the same as the specified value. |
| *sizeAbove* | 0..1 | The *contentSize* attribute of the *<contentInstance>* matched resource is equal to or greater than the specified value. |
| *sizeBelow* | 0..1 | The *contentSize* attribute of the *<contentInstance>* matched resource is smaller than the specified value. |
| *contentType* | 0..n | The *contentInfo* attribute of the *<contentInstance>* matched resource matches the specified value. |
| *attribute* | 0..n | This is an attribute of resource types (clause 9.6). Therefore, a real tag name is variable and depends on its usage and the value of the attribute can have wild card \*. E.g. *creator* of container resource type can be used as a filter criteria tag as "creator=Sam", "creator=Sam\*", "creator=\*Sam". |
| *childAttribute* | 0..n | A child of the matched resource meets the condition provided. The evaluation of this condition is similar to the *attribute* matching condition above. |
| *parentAttribute* | 0..n | The parent of the matched resource meets the condition provided. The evaluation of this condition is similar to the *attribute* matching condition above. |
| *semanticsFilter* | 0..n | Both semantic resource discovery and semantic query use *semanticsFilter* to specify a query statement that shall be specified in the SPARQL query language [5]. When a CSE receives a RETRIEVE request including a *semanticsFilter*, and the ***Semantic Query Indicator*** parameter is also present in the request, the request shall be processed as a semantic query; otherwise, the request shall be processed as a semantic resource discovery.  In the case of semantic resource discovery targeting a specific resource, if the semantic description contained in the <semanticDescriptor> of a child resource matches the semanticFilter, the URI of this child resource will be included in the semantic resource discovery result.  In the case of semantic query, given a received semantic query request and its query scope, the SPARQL query statement shall be executed over aggregated semantic information collected from the semantic resource(s) in the query scope and the produced output will be the result of this semantic query.  Examples for matching semantic filters in SPARQL to semantic descriptions can be found in [i.28]. |
| *filterOperation* | 0..1 | Indicates the logical operation (AND/OR) to be used for different condition tags. The default value is logical AND. |
| *contentFilterSyntax* | 0..1 | Indicates the Identifier for syntax to be applied for content-based discovery. |
| *contentFilterQuery* | 0..1 | The query string shall be specified when *contentFilterSyntax* parameter is present. |
| *operations* | 0..1(L) | A matched resource has a linked <accessControlPolicy> that grants the originator permission to perform the *operations* listed. |
| Filter Handling Conditions | | |
| *filterUsage* | 0..1 | Indicates how the filter criteria is used. If provided, possible values are 'discovery' and 'IPEOnDemandDiscovery'.  If this parameter is not provided, the Retrieve operation is a generic retrieve operation and the content of the child resources fitting the filter criteria is returned.  If *filterUsage* is 'discovery', the Retrieve operation is for resource discovery (clause 10.2.6), i.e. only the addresses of the child resources are returned.  If *filterUsage* is 'IPEOnDemandDiscovery', the other filter conditions are sent to the IPE as well as the discovery Originator ID. When the IPE successfully generates new resources matching with the conditions, then the resource address(es) shall be returned. This value shall only be valid for the Retrieve request targeting an <AE> resource that represents the IPE. |
| *limit* | 0..1 | The maximum number of resources to be included in the filtering result. This may be modified by the Hosting CSE. When it is modified, then the new value shall be smaller than the suggested value by the Originator. |
| *level* | 0..1 | The maximum level of resource tree that the Hosting CSE shall perform the operation starting from the target resource (i.e. ***To*** parameter). This shall only be applied for Retrieve operation. The level of the target resource itself is zero and the level of the direct children of the target is one. |
| *offset* | 0..1 | The number of direct child and descendant resources that a Hosting CSE shall skip over and not include within a Retrieve response when processing a Retrieve request to a targeted resource. |
| *applyRelativePath* | 0..1 | This attribute contains a resource tree relative path (e.g. ../tempContainer/LATEST). This condition applies after all the matching conditions have been used (i.e. a matching result has been obtained). The attribute determines the set of resource(s) in the final filtering result. The filtering result is computed by appending the relative path to the path(s) in the matching result. All resources whose Resource-IDs match that combined path(s) shall be returned in the filtering result. If the relative path does not represent a valid resource, the outcome is the same as if no match was found, i.e. there is no corresponding entry in the filtering result. |

The rules when multiple matching conditions are used together shall be as follows:

* Different condition tags shall use the "AND/OR" logical operation based on the *filterOperation* specified;

e.g. *createdBefore* = "time1" AND *unmodifiedSince* = "time2" if *filterOperation* = "AND" or "NULL", or *createdBefore* = "time1" OR *unmodifiedSince* = "time2" if *filterOperation* = "OR".

* Same condition tags shall use the "OR" logical operation, i.e. *filterOperation* doesn't apply to same conditions.

No mixed AND/OR filter operation will be supported.

Once the Request is delivered, the Receiver shall analyze the Request to determine the target resource.

If the target resource is addressing another M2M Node, the Receiver shall route the request appropriately.

If the target resource is addressing the Receiver, it shall:

* Check the existence of***To*** addressed resource.
* Identify the resource type by ***Resource Type***.
* Check the privileges for ***From*** Originator to perform the requested operation.
* Perform the requested operation (using ***Content*** content when provided) according to the provided request parameters as described above.
* Depending on the request result content, respond to the Originator with indication of successful or unsuccessful operation results. In some specific cases (e.g. limitation in the binding protocol or based on application indications), the Response could be avoided.

Table 8.1.2-3 summarizes the parameters specified in this clause for the Request message, showing any differences as applied to C, R, U, D or N operations. "M" indicates mandatory, "O" indicates optional, "N/A" indicates "not applicable".

Table 8.1.2-3: Summary of Request Message Parameters

| Request message parameter | | Operation | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| Create | Retrieve | Update | Delete | Notify |
| ***Mandatory*** | ***Operation*** - operation to be executed | M | M | M | M | M |
| ***To*** - the address of the target resource on the target CSE | M | M | M | M | M |
| ***From*** - the identifier of the message Originator | O  See note 1 | M | M | M | M |
| ***Request Identifier*** - uniquely identifies a Request message | M | M | M | M | M |
| ***Operation dependent*** | ***Content*** - to be transferred | M | O | M | N/A | M |
| ***Resource Type*** - of resource to be created | M | N/A | N/A | N/A | N/A |
| ***Optional*** | ***Originating Timestamp*** - when the message was built | O | O | O | O | O |
| ***Request Expiration Timestamp*** - when the request message expires | O | O | O | O | O |
| ***Result Expiration Timestamp*** - when the result message expires | O | O | O | O | O |
| ***Operational Execution Time*** - the time when the specified operation is to be executed by the target CSE | O | O | O | O | O |
| ***Response Type*** - type of response that shall be sent to the Originator | O | O | O | O | O |
| ***Result Persistence*** - the duration for which the reference containing the responses is to persist | O | O | O | O | N/A |
| ***Result Content*** - the expected components of the result | O | O | O | O | N/A |
| ***Event Category*** - indicates how and when the system should deliver the message | O | O | O | O | O |
| ***Delivery Aggregation*** - aggregation of requests to the same target CSE is to be used | O | O | O | O | O |
| ***Group Request Identifier*** - Identifier added to the group request that is to be fanned out to each member of the group | O | O | O | O | O |
| ***Group Request Target Members-***indicates subset of members of a group | O | O | O | O | N/A |
| ***Filter Criteria*** - conditions for filtered retrieve operation | N/A | O | O | O | N/A |
| ***Desired Identifier Result Type*** - format of resource identifiers returned | N/A | O | N/A | N/A | N/A |
| ***Token Request Indicator*** - indicating that the Originator may attempt Token Request procedure (for Dynamic Authorization) if initiated by the Receiver | O | O | O | O | O |
| ***Tokens*** - for use in dynamic authorization | O | O | O | O | O |
| ***Token IDs*** - for use in dynamic authorization | O | O | O | O | O |
| ***Role IDs*** - for use in role based access control | O | O | O | O | O |
| ***Local Token IDs*** - for use in dynamic authorization | O | O | O | O | O |
| ***Authorization Signature Indicator –*** for use in Authorization Relationship Mapping | O | O | O | O | N/A |
| ***Authorization Signature –*** for use in Authorization Relationship Mapping | O | O | O | O | N/A |
| ***Authorization Relationship Indicator -*** for use in Authorization Relationship Mapping | O | O | O | O | N/A |
| ***Semantic Query Indicator*** – for use in semantic queries | N/A | O | N/A | N/A | N/A |
| ***Release Version Indicator*** – the oneM2M release version that this request message conforms to. | M  See note 2 | M  See note 2 | M  See note 2 | M  See note 2 | M  See note 2 |
| ***Vendor Information*** | O | O | O | O | O |
| NOTE: 1.*From* parameter is optional in case of an AE CREATE request and mandatory for all other requests.  2. ***Release Version Indicator*** parameter is not present for the case when a request is targeting a Rel-1 entity and mandatory for all other cases. | | | | | | |

-------------------------------------------------- End of Change 4---------------------------------------------------