|  |
| --- |
|  |

|  |
| --- |
| CHANGE REQUEST |
| Meeting ID:\* | SDS #66 |
| Source:\* | Andreas Kraft, andreas.kraft@exactagss.com  |
| Date:\* | 2024-08-29 |
| Reason for Change/s:\* | Clarifying <schedule> child-resource procedure for <crossResourceSubscription> |
| CR against: Release\* | Rel 4 |
| CR against: WI\* | [ ]  Active <Work Item number> [x]  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-0004, v.4.21.0 |
| Clauses \* | 7.5.1.2.2 |
| Type of change: \* | [ ]  Editorial change[x]  Bug Fix or Correction[ ]  Change to existing feature or functionality[ ]  New feature or functionalityOnly ONE of the above shall be ticked |
| Other TS/TR(s) impacted |  |
| 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 2020 (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.

If this is a correction, and the change applies to previous releases, a separate “mirror CR” should be posted at the same time as 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. Include any changes to references, definitions, and abbreviations in the same deliverable.

Follow the drafting rules.

All pictures must be editable.

Check spelling and grammar.

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 proposed new clause is 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 the 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

The CR proposes to add the <crossResourceSubscription> resource type to “7.5.1.2.2 Notification for <subscription> resources”, step 3.0.

The <schedule> resource is a child resource of the <crossResourceSubscription>, but the behaviour for it is not described. Since the effect, ie. creating notifications under certain circumstances, is the same as for <subscription>, the proposed change is just to add the <crossResourceSubscription> to the text of 7.5.1.2.2.

When it comes to the notification handling the <crossResourceSubscription> procedures are pointing to the respective <subscription> procedures in the specifications.

### -----------------------Start of change 1-------------------------------------------

##### 7.5.1.2.2 Notification for <subscription> resources

When the notification message is forwarded or aggregated by transit CSEs, the Originator or a transit CSE shall check whether there are notification policies to enforce between subscription resource Hosting CSE and the notification target. In that case, the transit CSE as well as the Originator shall process Notify request primitive(s) by using the corresponding policy and send processed Notify request primitive(s) to the next CSE with notification policies related to the enforcement so that the transit CSE is able to enforce the policy defined by the subscriber. The notification policies related to the enforcement at this time is verified by using the subscription reference in the Notify request primitive. In the notification policies, the *latestNotify* attribute is only enforced in the transit CSE as well as the Originator.

If ***Event Category*** parameter is set to "latest" in the notification request primitive, the transit CSE as well as Originator shall cache the most recent Notify request. That is, if a new Notify request is received by the CSE with a subscription reference that has already been buffered for a pending Notify request, the newer Notify request will replace the buffered older Notify request.

***Originator:***

When an event is generated, the Originator shall execute the following steps in order:

Step 1.0 Check the *eventNotificationCriteria* attribute of the <subscription> resource associated with the modified resource:

* If the *eventNotificationCriteria* attribute is set, then the Originator shall check whether the corresponding event matches with the event criteria. If multiple matching conditions of different types (i.e. different condition tags) are present in the *eventNotificationCriteria* attribute, then the combined condition shall be derived by applying the logical operation specified by the *filterOperation* condition. By default the logical AND operation shall be used if the *filterOperation* condition is not present.
* If *notificationEventType* is not set within the *eventNotificationCriteria* attribute and the *operationMonitor* is also not present, the Originator shall use the default setting of "Update\_of\_Resource" to compare against the event.
* If the *notificationEventType* has the value "Create\_of\_Direct\_Child\_Resource" or "Delete of Direct Child Resource" and the *childResourceType* condition is also present, then the matching event shall only be detected if one of the child resource types present in the list has been created or deleted, respectively. If the *childResourceType* condition is not present then a matching event is generated whenever any child resource is created or deleted.
* If the *notificationEventType* has either an explicit or default value of "Update\_of\_Resource" and the *attribute* condition is also present then the matching event shall only be detected if one of the attributes in the list has been updated. If the *attribute* condition is not present then a matching event is generated whenever any attribute has been updated.
* If the event matches, go to the step 2.0. Otherwise, the Originator shall discard the corresponding event.
* If the *eventNotificationCriteria* attribute is not configured, the Originator shall use the default setting of "Update\_of\_Resource" for the *notificationEventType* and then continue with the step 2.0.

Step 2.0 The Originator shall check the notification policy as described in the below steps, but the notification policy may be checked in different order. After checking the notification policy in step 2.0 (i.e. from step 2.1 to step 2.6), then continue with step 3.0.

Step 2.1 The Originator shall determine the type of the notification per the *notificationContentType* attribute. The possible values of for *notificationContentType* attribute are "Modified Attributes", "All Attributes", "ResourceID", "Trigger Payload" or “TimeSeries notification”. This attribute may be used jointly with the *notification*EventType attribute in the eventNotificationCriteria to determine if it is the attributes/resourceID of the subscribed-to resource or the attributes/resourceID of the child resource of the subscribed-to resource that shall be returned in the content of the notification:

* If the value of *notificationContentType* is set to "Modified Attributes", the Notify request primitive shall include the partial resource containing modified attribute(s) only (Refer to clause 7.2.1.2 for response content description).
* If the value of *notificationContentType* is set to "All Attributes", the Notify request primitive shall include the complete resource with all attributes (Refer to clause 7.2.1.2 for response content description).
* If the value of *notificationContentType* is set to "ResourceID", the Notify request primitive shall include the URI of the resource (Refer to clause 7.2.1.2 for response content description).
* If the value of *notificationContentType* is set to "Trigger Payload", the Notify request primitive shall include the trigger payload (Refer to clause 9.2.1 for trigger payload description).
* If the value of *notificationContentType* is set to "TimeSeries notification", the Notify request primitive shall include a timeSeriesNotification (Refer to clause 6.3.5.69 for timeSeriesNotification description).

In addition to the procedure described above, if the *primitiveProfileID* attribute of the <subscription> resource is configured with the resource identifier of a <primitiveProfile> resource, then the Originator shall apply the <primitiveProfile> resource to the request parameters of the Notify request primitive by adding, replacing or deleting any applicable request parameters defined in the *additions* or *deletions* attributes of the referenced <primitiveProfile> resource.

If *notificationContentType* is set to "Modified Attributes" or "All Attributes", the Originator shall also apply the referenced <primitiveProfile> resource to the representation of the subscribed-to resource or the child resource of the subscribed-to resource that is included in the content of the notification, if applicable. Before doing this the Originator shall make the following checks:

* Check that the primitive profile’s *resourceTypes* and *resourceIDs* attributes match the subscribed-to resource or the child resource of the subscribed-to resource that is included in the content of the notification.
* Check that the primitive profile’s *operations* attribute includes the Notify operation and the *releaseVersions* attribute matches the release version indicator of the notification.
* Check that the *applicability* attribute includes the value “NOTIFICATIONS\_FROM\_CSE”.

If all these checks are successful, the Originator shall apply the <primitiveProfile> resource to the resource attributes included in the content of the notification by adding, replacing or deleting resource attributes defined in the *additions* and *deletions* attribute of the <primitiveProfile> resource.

Step 2.2 Check the *notificationEventCat* attribute:

* If the *notificationEventCat* attribute is set, the Notify request primitive shall employ the ***Event Category*** parameter as given in the *notificationEventCat* attribute. Then continue with the step 2.3.
* If the *notificationEventCat* attribute is not configured, then continue with step 2.3.

Step 2.3 Check the *latestNotify* attribute:

* If the *latestNotify* attribute is set, the Originator shall assign ***Event Category*** parameter of value "latest" of the notifications generated pertaining to the subscription created.

Step 2.4 Check the batching notifications policy and the *rateLimit* attribute:

* See details in oneM2M TS-0001 [6], clause 10.2.10.7.
* If both the *batchNotify* and *primitiveProfileID* attributes of the <subscription> resource are configured, the Originator shall attempt to apply the referenced <primitiveProfile> resource to the individual notifications embedded within an aggregated notification using the same procedure as described in Step 2.1. In addition, the Originator shall also apply the primitive profile to the request parameters of the aggregated notification request primitive by adding, replacing or deleting any applicable request parameters defined in the *additions* or *deletions* attributes of the referenced <primitiveProfile> resource

NOTE: The use of some attributes such as *preSubscriptionNotify* is not supported in the present document.

Step 2.5 Check the *notificationURI* attribute:

* The Originator shall fetch the *notificationURI* attribute and set the value to the ***To*** parameter of the Notify request. When the *notificationURI* attribute contains more than one target, the Originator shall generate each Notify request per target.
* If the *notificationURI* attribute includes the notification serialization indication, in form of key-value pair, e.g. "ct=json", after the delimiter "?", the Originator shall serialize the notification for the notification target in that serialization type. The delimiter with the serialization indication shall be removed when the target is set to the ***To*** parameter of the Notify request. Then continue with step 3.0.

Step 3.0 The Originator shall check the notification and reachability schedules, but the notification schedules may be checked in different order:

* If the <subscription> resource associated with the modified resource or if the <crossResourceSubscription> resource includes a <notificationSchedule> child resource, the Originator shall check the time periods given in the scheduleElement attribute of the <notificationSchedule> child resource.
* Also, the Originator shall check the reachability schedule associated with the Receiver by exploring its <schedule> resource. If reachability schedules are not present in a Node then that Entity is considered to be always reachable.
* If notificationSchedule and reachability schedule indicate that message transmission is allowed, then proceed with step 5.0. Otherwise, proceed with step 4.0.
* In particular, if the *notificationEventCat* attribute is set to 'immediate' and the <notificationSchedule> resource does not allow transmission, then go to step 5.0 and send the corresponding Notify request primitive by temporarily ignoring the Originator's notification schedule.

Step 4.0 Check the *pendingNotification* attribute:

* If the *pendingNotification* attribute is set, then the Originator shall cache pending Notify request primitives according to the *pendingNotification* attribute. The possible values are 'sendLatest' and 'sendAllPending'. If the value of pendingNotification is set to 'sendLatest', the most recent Notify request primitive shall be cached by the Originator and it shall set the ***Event Category*** parameter to "latest". If *pendingNotification* is set to 'sendAllPending', all Notify request primitives shall be cached by the Originator. If the *pendingNotification* attribute is not configured, the Originator shall discard the corresponding Notify request primitive. Any cached Notify request primitives are sent to the Receiver once message transmission becomes possible (see the step 6.0).

Step 5.0 Check the *expirationCounter* attribute:

* If the *expirationCounter* attribute is set, then it shall be decreased by one when the Originator successfully sends the Notify request primitive. If the counter equals to zero('0'), the corresponding <subscription> resource shall be deleted. Then end the 'Compose Notify Request Primitive' procedure.
* If the *expirationCounter* attribute is not configured, then end the 'Compose Notify Request Primitive' procedure.

When message transmission becomes possible, the Originator shall execute the following steps in order:

Step 6.0 If the *pendingNotification* attribute is set, the Originator shall send any cached Notify request primitives and then continue with the step 7.0

Step 7.0 Check the *expirationCounter* attribute:

* If the *expirationCounter* attribute is set, then its value shall be decreased by one when the Originator successfully sends the Notify request primitive. If the counter meets zero, the corresponding <subscription> resource shall be deleted. Then end the 'Compose Notify Request Primitive' procedure.
* If the *expirationCounter* attribute is not configured, then end the 'Compose Notify Request Primitive' procedure.

***Receiver:***

When the Hosting CSE receives a Notify request primitive, the Hosting CSE shall check validity of the primitive parameters. In case the Receiver is a transit CSE which forwards or aggregates Notify request primitives before sending to the subscriber or other transit CSEs, upon receiving the Notify request primitive with the ***Event Category*** parameter set to "latest", the Receiver shall identify the latest Notify request primitive with the same subscription reference while storing Notify request primitives locally. When the Receiver as a transit CSE needs to send pending Notify request primitives, it shall send the latest Notify request primitive. When the Receiver as a transit CSE needs to send Notify request primitives, it shall use one of the serializations specified in the subscriber or other transit CSE *contentSerialization* attribute. If there is no *contentSerialization* value specified the transit CSE may use any serialization format.

### -----------------------End of change 1---------------------------------------------

### -----------------------Start of change 2---------------------------------------------

### -----------------------End of change 2---------------------------------------------

### -----------------------Start of change 3---------------------------------------------

### -----------------------End of change 3---------------------------------------------

### -----------------------Start of change 4---------------------------------------------

### -----------------------End of change 4---------------------------------------------

### -----------------------Start of change 5---------------------------------------------

### 9.6.35 Resource Type *flexContainer*

…………….

The *<flexContainer>* resource shall contain the attributes specified in table 9.6.35-2.

Table 9.6.35-2: Attributes of <*flexContainer*> resource

| **Attributes of *<flexContainer>*** | **Multiplicity** | **RW/****RO/****WO** | **Description** | ***<flexContainerAnnc>* 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* | 0..1 (note) | RW | See clause 9.6.1.3. | MA |
| *accessControlPolicyIDs* | 0..1 (L) | RW | See clause 9.6.1.3. | MA |
| *subscriptionIDs* | 0..1 (L) | RW | See clause 9.6.1.3. | MA |
| *labels* | 0..1 (L) | RW | See clause 9.6.1.3. | MA |
| *creationTime* | 0..1(note) | RO | See clause 9.6.1.3. | NA |
| *lastModifiedTime* | 0..1(note) | RO | See clause 9.6.1.3. | NA |
| *stateTag* | 1 | RO | See clause 9.6.1.3.This *stateTag* attribute value shall be incremented when a custom attribute of the flexContainer is modified. | NA |
| *announceTo* | 0..1 (L) | RW | See clause 9.6.1.3. | NA |
| *announcedAttribute* | 0..1 (L) | RW | See clause 9.6.1.3. | NA |
| *announceSyncType* | 0..1 | RW | See clause 9.6.1.3. | MA |
| *dynamicAuthorizationConsultationIDs* | 0..1 (L) | RW | See clause 9.6.1.3. | OA |
| *creator* | 0..1 | RO | See clause 9.6.1.3. | NA |
| *custodian* | 0..1 | RW | See clause 9.6.1.3. | NA |
| *location* | 0..1 | RW | See clause 9.6.1.3. | OA |
| *fcinEnabled* | 0..1 | RW | Controls the creation of <flexContainerInstance> child resources. Valid values are “TRUE” and “FALSE”.  | OA |
| *maxNrOfInstances* | 0..1 | RW | Maximum number of direct child *<flexContainerInstance>* resources in the *<flexContainer>* resource. | OA |
| *maxByteSize*  | 0..1 | RW | Maximum size in bytes of custom attributes that is allocated for the *<flexContainer>* resource for all direct child *<flexContainerInstance>* resources. | OA |
| *maxInstanceAge* | 0..1 | RW | Maximum age of a direct child *<flexContainerInstance>* resources in the *<flexContainer>* resource. The value is expressed in seconds. | OA |
| *currentNrOfInstances* | 0..1 | RO |  Current number of direct child *<flexContainerInstance>* resources in the *<flexContainer>* resource. It is limited by the maxNrOfInstances. The currentNrOfInstances attribute of the *<flexContainer>* resource shall be updated on successful creation or deletion of direct child *<flexContainerInstance>* resource of *<flexContainer>* resource. | OA |
| *currentByteSize*  | 0..1 | RO | Current size in bytes of custom attributes stored in all direct child *<flexContainerInstance>* resources of the *<flexContainer>* resource. It is limited by the maxByteSize. The currentByteSize attribute of the *<flexContainer>* resource shall be updated on successful creation or deletion of a direct child *<flexContainerInstance>* resource of *<flexContainer>* resource. | OA |
| *containerDefinition* | 1 | WO | This contains an identifier reference (URI) to the <*flexContainer*> schema definition which shall be used by the CSE to validate the syntax of the <*flexContainer*> resource.This URI may refer to one of the oneM2M <*flexContainer*> definitions specified in the following documents:* Generic Interworking [6]
* AllJoyn Interworking [7]
* Home Domain Information Model [8]

A list of oneM2M <*flexContainer*> definitions is also provided in clause 9.6.1.2.2 of oneM2M TS‑0004 [3].Other URI for other *<flexContainer>* definitions may be specified. | MA |
| *ontologyRef* | 0..1 | RW | A reference (URI) of the ontology used to represent the information that is stored in the present *<flexContainer>* resource. | OA |
| *contentSize* | 1 | RO | Sum of the size in bytes of all of the custom attributes. | NA |
| *nodeLink* | 0..1 | RW | The resource identifier of a <node> resource that stores the node specific information of the NoDN on which the interworked service represented by this <flexContainer> resource resides. | OA |
| *[customAttribute]* | 0..n | RW | Specialization-specific attribute(s). Name and data type defined in each specialization of <*flexContainer>* resource. | OA |
| NOTE: When an instance of <*flexContainer*> is a child of a <*flexContainer*> resource, these attributes can be optional. Their presence is determined by the respective definition referred to by the *containerDefinition* attribute. |

When a new <*flexContainerInstance*> child resource is created, the attributes in Table 9.6.35-3 shall be copied from the parent <*flexContainer*> resource to the new <*flexContainerInstance*> resource.

Table 9.6.35-3: Copied attributes of the parent <*flexContainer*> resource

| Attributes of *<flexContainer>* | Description |
| --- | --- |
| *fcinEnabled* |  |
| *labels* |  |
| *stateTag* |  |
| *[customAttribute]* | All custom attributes are copied |

### -----------------------End of change 5---------------------------------------------

### -----------------------Start of change 6---------------------------------------------

### 9.6.36 Resource Type *timeSeries*

The *<timeSeries>* resource represents a container for Time Series Data instances. It is used to share information with other entities and potentially to track, detect and report the missing data in Time Series. A *<timeSeries>* resource has no associated content. It has only attributes and child resources.

Table 9.6.36-1: Child resources of <*timeSeries*> resource

| Child Resources of *<timeSeries>* | Child Resource Type | Multiplicity | Description | *<timeSeriesAnnc>* Child Resource Types |
| --- | --- | --- | --- | --- |
| *[variable]* | *<semanticDescriptor>* | 0..n | See clause 9.6.30 | *<semanticDescriptor>, <semanticDescriptorAnnc>* |
| *[variable]* | *<timeSeriesInstance>* | 0..n | See clause 9.6.37 | *<timeSeriesInstance>,**<timeSeriesInstanceAnnc>* |
| *[variable]* | *<subscription>* | 0..n | See clause 9.6.8 | *<subscription>* |
| *la* | *<latest>* | 1 | See clause 9.6.27 | *None* |
| *ol* | *<oldest>* | 1 | See clause 9.6.28 | *None* |
| *[variable]* | *<transaction>* | 0..n | See clause 9.6.48 | *<transaction>* |
| *[variable]* | *<action>* | 0..n | See clause 9.6.61 | *<actionAnnc>* |

The *<timeSeries>* resource shall contain the attributes specified in table 9.6.36-2.

Table 9.6.36-2: Attributes of <*timeSeries*> resource

| Attributes of *<timeSeries>* | Multiplicity | RW/RO/WO | Description | *<timeSeriesAnnc>* 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 |
| *subscriptionIDs* | 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 |
| *announceTo* | 0..1 (L) | RW | See clause 9.6.1.3. | NA |
| *announcedAttribute* | 0..1 (L) | RW | See clause 9.6.1.3. | NA |
| *announceSyncType* | 0..1 | RW | See clause 9.6.1.3. | MA |
| *dynamicAuthorizationConsultationIDs* | 0..1 (L) | RW | See clause 9.6.1.3. | OA |
| *creator* | 0..1 | RO | See clause 9.6.1.3. | NA |
| *custodian* | 0..1 | RW | See clause 9.6.1.3. | NA |
| *location* | 0..1 | RW | See clause 9.6.1.3. | OA |
| *maxNrOfInstances* | 0..1 | RW | Maximum number of direct child *<timeSeriesInstance>* resources in the <*timeSeries*> resource. | OA |
| *maxByteSize* | 0..1 | RW | Maximum size in bytes of data that is allocated for the *<timeSeries>* resource for all direct child*<timeSeriesInstance>* resources. | OA |
| *maxInstanceAge* | 0..1 | RW | Maximum age of a direct child *<timeSeriesInstance>* resource in the <*timeSeries*> *resource*. The value is expressed in seconds. | OA |
| *currentNrOfInstances* | 1 | RO | Current number of direct child *<timeSeriesInstance>* resource in the <*timeSeries*> resource. It is limited by the *maxNrOfInstances*. The *currentNrOfInstances* attribute of the <timeSeries> resource shall be updated on successful creation or deletion of direct child < *timeSeriesInstance* > resource of <timeSeries > resource. | NA |
| *currentByteSize* | 1 | RO | Current size in bytes of data stored in all direct child *<timeSeriesInstance>* resources of a <*timeSeries*> resource. It is limited by the *maxByteSize*. The *currentByteSize* attribute of the <timeSeries> resource shall be updated on successful creation or deletion of direct child < *timeSeriesInstance* > resource of <timeSeries > resource. | NA |
| *periodicInterval* | 0..1 | RW | If the Time Series Data is periodic, this attribute shall contain the expected amount of time between two instances of Time Series Data. | OA |
| *periodicIntervalDelta* | 0..1 | RW | If the Time Series Data is periodic, this attribute contains a +/- delta value relative to *periodicInterval* for the purpose of detecting missing data.The value of this attribute shall be less than or equal to (*periodicInterval/2*).If the attribute is omitted the hosting CSE can use a local policy to determine a default value. | OA |
| *missingDataDetect* | 1 | RW | Indicates whether the Receiver shall detect the missing Time Series Data if it is periodic. The default value is false. | NA |
| *ontologyRef* | 0..1 | RW | A reference (URI) of the ontology used to represent the information that is stored in the child *<timeSeriesInstance>* resources of the present *<timeSeriesData>* resource (see note). | OA |
| *missingDataMaxNr* | 0..1 | RW | Maximum number of entries in the *missingDataList* if the *periodicInterval* is set and the *missingDataDetect* is TRUE. | OA |
| *missingDataList* | 0..1(L) | RO | The list of the *dataGenerationTime* valuerepresenting the missing Time Series Data in descending order by time if the *periodicInterva*l is set and the *missingDataDetect* is TRUE. | NA |
| *missingDataCurrentNr* | 1 | RO | Current number of the missing Time Series Data in the *missingDataList*. | NA |
| *missingDataDetectTimer* | 0..1 | RW | The *missingDataDetectTimer* is a duration after which a <*timeSeriesInstance*> shall be considered missing by the hosting CSE.If *periodicIntervalDelta* is present, the value of this attribute shall be greater than *periodicIntervalDelta.* | OA |
| *contentInfo* | 0..1 | WO | This attribute contains information to understand the contents of the *content* attribute of <timeSeriesInstance>. It shall be composed of two mandatory components consisting of an Internet Media Type (as defined in the IETF RFC 6838 [i.36]) and an encoding type. In addition, an optional content security component may also be included. The format of this attribute is defined in oneM2M TS‑0004 [3].This attribute should be used to represent the content information of the *content* attribute of child <*timeSeriesInstance*> resources so that AEs can understand the content. | OA |
| NOTE: The access to this URI is out of scope of oneM2M. |

### -----------------------End of change 6---------------------------------------------

### -----------------------Start of change 7---------------------------------------------

### 9.6.15 Resource Type *mgmtObj*

The *<mgmtObj>* resource contains management data which represents individual M2M management functions. It represents a general structure to map to technology specific data model e.g. OMA DM [i.3], BBF TR-069 [i.2] and LWM2M [i.4]. Each instance of *<mgmtObj>* resource shall be mapped to single technology specific protocol.

The *<mgmtObj>* resource shall contain the child resource specified in table 9.6.15-1.

Table 9.6.15-1: Child resources of *<mgmtObj>* resource

| Child Resources of *<mgmtObj>* | Child Resource Type | Multiplicity | Description | *<mgmtObjAnnc>* Child Resource Type |
| --- | --- | --- | --- | --- |
| *[variable]* | *<subscription>* | 0..n | See clause 9.6.8 | *<subscription>* |
| *[variable]* | *<semanticDescriptor>* | 0..n | See clause 9.6.30 | *<semanticDescriptor>, <semanticDescriptorAnnc>* |
| *[variable]* | *<transaction>* | 0..n | See clause 9.6.48 | *<transaction>* |

The *<mgmtObj>* resource shall contain the attributes specified in table 9.6.15-2.

Table 9.6.15-2: Attributes of *<mgmtObj>* resource

| Attributes of *<mgmtObj>* | Multiplicity | RW/RO/WO | Description | *<mgmtObjAnnc>* 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 |
| *subscriptionIDs* | 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 |
| *labels* | 0..1 (L) | RW | See clause 9.6.1.3. | MA |
| *announceTo* | 0..1 (L) | RW | See clause 9.6.1.3. | NA |
| *announcedAttribute* | 0..1 (L) | RW | See clause 9.6.1.3. | NA |
| *announceSyncType* | 0..1 | RW | See clause 9.6.1.3. | MA |
| *dynamicAuthorizationConsultationIDs* | 0..1 (L) | RW | See clause 9.6.1.3. | OA |
| *custodian* | 0..1 | RW | See clause 9.6.1.3. | NA |
| *mgmtDefinition* | 1 | WO | Specifies the type of *<mgmtObj>* resource e.g. software, firmware, memory. The list of the value of the attribute can be seen in annex D. | MA |
| *mgmtSchema* | 0..1 | WO | Contains a URI to the <*mgmtObj*> schema definition which shall be used by the Hosting CSE to validate the syntax of incoming primitives targeting this <*mgmtObj*> resource.This URI may refer to a oneM2M specified <*mgmtObj*> definition as well as other <mgmtObj> definitions. | MA |
| *objectIDs* | 0..1 (L) | WO | Contains the list URNs that uniquely identify the technology specific data model objects used for this *<mgmtObj>* resource as well as the managed function and version it represents. This attribute shall be provided during the creation of the *<mgmtObj>* resource and shall not be modifiable afterwards.If the *<mgmtObj>* resource is mapped to multiple technology specific data model objects, this attribute shall list all URNs for each mapped technology specific data model objects. This is mandatory for the *<mgmtObj>*, for which the data model is not specified by oneM2M but mapped from technology specific data model. | OA |
| *objectPaths* | 0..1 (L) | WO | Contains the list of local paths of the technology specific data model objects on the managed entity which is represented by the *<mgmtObj>* resource in the Hosting CSE.This attribute shall be provided during the creation of the *<mgmtObj>*, so that the Hosting CSE can correlate the created *<mgmtObj>* with the technology specific data model object on the managed entity for further management operations. It shall not be modifiable after creation.The format of this attribute shall be a local technology specific data model object path in the form as specified by technology specific protocol. (e.g. "./anyPath/Fw1" in OMA DM [i.3], "Device.USBHosts.Host.3." in BBF TR‑069 [i.2]).The combination of the *objectPaths* and the *objectIDs* attribute, allows to address the technology specific data model. | OA |
| *mgmtLink* | 0..1 (L) | RW | This attribute contains reference to a list of other *<mgmtObj>* resources in case a hierarchy of *<mgmtObj>* is needed. | OA |
| *[objectAttribute]* | 0..n | RW | Each *[objectAttribute]* is mapped from a leaf node of a hierarchical structured technology specific data model object (including oneM2M data model and the technology specific data model objects) based on the mapping rules below the table. | OA |
| *description* | 0..1 | RW | Text format description of <*mgmtObj>*. | OA |

When mapping objects from technology specific protocol to a corresponding *<mgmtObj>* resource, the following rules shall apply:

* The root objects of technology specific data model objects maps to the *<mgmtObj>* resource.
* For the child of the root of technology specific data model objects:
* **Rule 1:** If the child technology specific data model object cannot have another child technology specific data model object, the technology specific data model object maps to the *[objectAttribute]* attribute of the *<mgmtObj>* resource with the same resource name.
* **Rule 2:** If the child technology specific data model object can have another child technology specific data model object, the technology specific data model object maps to a new *<mgmtObj>* resource. The ID of the new *<mgmtObj>* resource is stored as an *mgmtLink* attribute of the *<mgmtObj>* resource which is mapped from the parent technology specific data model object.

### -----------------------End of change 7---------------------------------------------

CHECK LIST

* Does this Change Request include an informative introduction containing the problem(s) being solved, and a summary list of proposals.?
* Does this CR contain changes related to only one particular issue/problem?
* Have any mirror CRs been posted?
* Does this Change Request make **all** the changes necessary to address the issue or problem? E.g. A change impacting 5 tables should not include a proposal to change only 3 tables?Does this Change Request follow the drafting rules?
* Are all pictures editable?
* Have you checked the spelling and grammar?
* Have you used change bars for all modifications?
* Does the change 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.)
* Are multiple changes in this CR 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.?