|  |
| --- |
|  |

|  |  |
| --- | --- |
| CHANGE REQUEST | |
| Meeting ID:\* | SDS #67 |
| Source:\* | Andreas Kraft, Exacta, [andreas.kraft@exactagss.com](mailto:andreas.kraft@exactagss.com)  Poornima Shandilya, C-DOT, [poornima@cdot.in](mailto:poornima@cdot.in) |
| Date:\* | 2024-07-31 |
| Reason for Change/s:\* | TS-0004: Clarification for setting the notification’s creator attribute |
| CR against: Release\* | Release 4 |
| CR against: WI\* | Active WI-xxxx  MNT maintenance / < Work Item number(optional)>  Is this a mirror CR? Yes  No  mirror CR number: (Note to Rapporteur - use latest agreed revision)  STE Small Technical Enhancements / < Work Item number (optional)>  Only ONE of the above shall be ticked |
| CR against: TS/TR\* | TS-0004 v.4.20.0 |
| Clauses \* | 7.5.1.2.2, 7.5.1.2.4, 7.5.1.2.9, 7.5.1.2.19 |
| Type of change: \* | Editorial change  Bug Fix or Correction  Change to existing feature or functionality  New feature or functionality  Only ONE of the above shall be ticked |
| Impacted other TS/TR(s) |  |
| Post Freeze checking:\* | This CR contains only essential changes and corrections? YES  NO  This CR may break backwards compatibility with the last approved version of the TS? YES  NO |
| Template Version: January 2017 (Do not modify) | |

**oneM2M Notice**

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

GUIDELINES for Change Requests:

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

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

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

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

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

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

Follow the drafting rules.

All pictures must be editable.

Check spelling and grammar to the extent practicable.

Use Change bars for modifications.

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

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

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

Introduction

This Change Request proposes a change to <subscription> notification procedures in TS-0004, 7.5.1.2.x .

In TS-0001, clause 10.2.10.7, the procedure specifies when the *creator* element of a Notify request is present, ie.

“When there is an event for a <subscription> resource that triggers a notification, the <subscription> Hosting CSE shall include in the notification the creator if the <subscription> resource has creator attribute.”

This part of the procedure is currently not present in the relevant clauses in TS-0004.

If this CR is accepted, further CRs will be submitted for TS-0004 R3 and R2 (the *creator* element is defined for “m2m:notification” in these releases).

### \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 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.

In addition, if the <subscription> resource has a *creator* attribute, then the Originator shall set the *creator* element of the Notify request primitive to the value of the <subscription> resource’s *creator* attribute.

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 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 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

##### 7.5.1.2.4 Notification of Subscription Deletion

***Originator:***

When a <subscription> resource is deleted the Originator shall send Notify request primitives to the <subscription> resource's subscriberURI and *associatedCrossResourceSub* if they are configured:

a) The *subscriptionDeletion* element of the notification data object set as true.

b) The *subscriptionReference* element of the notification data object set as the resource identifier of the <subscription> resource.

c) The ***To*** parameter shall be set to the entity indicated in *subscriberURI* or *associatedCrossResourceSub*.

d) If the <subscription> resource has a *creator* attribute, then the Originator shall set the *creator* element of the notification to the value of the <subscription> resource’s *creator* attribute. When a <subscription> is not itself deleted but a target is removed from its *associatedCrossResourceSub* attribute, the Originator shall send a Notify request primitive to that target as follows:

a) The *subscriptionDeletion* element of the notification data object set as true.

b) The *subscriptionReference* element of the notification data object set as the resource identifier of the <subscription> resource.

c) The ***To*** parameter shall be set to the target removed from *associatedCrossResourceSub.*

d) If the <subscription> resource has a *creator* attribute, then the Originator shall set the *creator* element of the notification to the value of the <subscription> resource’s *creator* attribute.

***Receiver:***

When a Hosting CSE receives a subscription deletion notification targeted to an existing <crossResourceSubscription> resource then the <crossResourceSubscription> resource is deleted using the procedure defined in clause 7.4.58.2.4 except when the crossResourceSubscription resource delete procedure is already in process (so as to avoid an endless loop).

### \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of Change 2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Start of Change 3 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

##### 7.5.1.2.9 Notification for missing Time Series Data

When an AE wants to be informed of the number of missing data points in a given renewable time duration, the AE shall request the creation of a *<subscription>* resource and set the *missingData* in the *eventNotificationCriteria* conditions to specify the reporting policy. The *notificationEventType* element in the eventNotificationCriteria shall have a value of “Report on missing data points”. This enables the AE to keep track of the number of missing data points and their corresponding time-stamps over a predefined but renewable duration (the "window duration" of the *missingData*).

***Originator(Hosting CSE):***

No change from the procedures in clause 7.5.1.2.2. except the following addition in Step 1.0:

When the first missing data point is detected (i.e. a detection of the first discontinuous time-stamp), following the creation of the subscription, the Hosting CSE shall start a timer associated with that subscription and start counting the number of the missing data points. The timer is set according to the *duration* in the subscription’s *missingData* condition. The reporting policy is governed by the rules below:

* If the total number of missing data points becomes equal to the *number* specified in the *missingData* conditionbefore the timer expires, a NOTIFY request shall be sent with a *timeSeriesNotification* element in the *notificationEvent*/*representation* element of the notification If the <subscription> resource has the *creator* attribute set, then the *creator* element of the notification shall be set to the value of the <subscription> resource’s *creator* attribute. The missing data points counter shall continue counting while the timer continues to run (since it did not expire). A similar NOTIFY request shall be sent for each subsequent missing data point detected until the timer expires (see next bullet for behaviour when the timer expires).
* If the timer expires the missing data points counter is reset back to 0. The timer is restarted upon subsequent detection of missing data.
* The reset of the timer and the missing data points counter upon timer expiry shall continue until such time as the subscription is cancelled or terminated.
* If no missing data points have been detected at all during the lifetime of a subscription, then no timer shall be started at all. But once a timer is started triggered by the first missing data point, then the above rules in the previous bullets shall apply.

No change for the remaining steps from the procedures in clause 7.5.1.2.2.

### \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of Change 3 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Start of Change 4 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

##### 7.5.1.2.19 Notification for Subscription Blocking Triggered update

Whenever the Hosting CSE receives an update request primitive for a target resource which has a <subscription> resource with *notificationEventType* set to "Blocking\_Update", it shall perform the steps listed below before Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation” is performed.

1. If the *attribute* condition tag of the *eventNotificationCriteria* attribute of the <subscription> resource is set, check that the *attribute* condition tag matches the modified attributes in the received UPDATE request.
2. Prevent or block all other UPDATE request primitives to this target resource.
3. Create a Notification request primitive and configure the request parameters as follows.
4. Set the *representation* attribute of the notification to the representation of the target resource contained in the received UPDATE request primitive.
5. If the <subscription> resource has a *creator* attribute, then the Originator shall set the *creator* element of the notification to the value of the <subscription> resource’s *creator* attribute.Send the Notification request primitive to the target specified in *notificationURI.*
6. Wait for a Notification response.
7. Process the Notification response primitive

a) If the notification ***Response Status Code*** is not successful, return a response to the original blocked UPDATE request primitive with a ***Response Status Code*** according to clause 7.3.2.9 .

b) If the notification ***Response Status Code*** is successful, perform Recv-6.5 "Create/Update/Retrieve/Delete/Notify operation” for the received UPDATE request.

1. Allow all other UPDATE request primitives for this target resource.

### \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of Change 4 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*