|  |
| --- |
|  |

|  |
| --- |
| CHANGE REQUEST |
| Meeting ID:\* | ARC 36 |
| Source:\* | Bob Flynn, Convida Wireless, Flynn.Bob@ConvidaWireless.com Dale Seed, Convida Wireless, Seed.Dale@ConvidaWireless.com |
| Date:\* | 2018-07-08 |
| Reason for Change/s:\* | See the introduction  |
| CR against: Release\* | Release 3 |
| CR against: WI\* | [x]  Active - WI-0058 - 3GPP & Cellular IoT Interworking [ ]  MNT maintenance / < Work Item number(optional)>Is this a mirror CR? Yes [ ]  No [x] 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 Version 3.11.0 |
| Clauses \* | 9.6.8,10.2.10.2, 10.2.10.7 |
| Type of change: \* | [ ]  Editorial change[ ]  Bug Fix or Correction[x]  Change to existing feature or functionality[ ]  New feature or functionalityOnly ONE of the above shall be ticked |
| Impacted other TS/TR(s) | <TS/TR number>, <Version Number>, and <Description on which aspect should be reflected in this TS/TR> |
| Post Freeze checking:\* | This CR contains only essential changes and corrections? YES [x]  NO [ ] This CR may break backwards compatibility with the last approved version of the TS? YES [ ]  NO [x]  |
| 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

### R01 – Changes discussed during review.

The blocking UPDATE functionality allows the redirection of an UPDATE request to another entity (AE or CSE).

Since this is enabled through the use of a subscription, which has a wide variety of attributes to support other uses cases, further definition is needed to describe how this type of subscription should/can be created.

Change 1 –

Reword the description of “blocking subscriptions” in the introduction by providing a general description and restructuring a couple paragraphs.

Describe that only a single blocking subscription to a resource SHALL exist.

Describe that the notification content shall only include the attributes that are modified

Change 2 –

Update procedure to check that only a single subscription exists.

Change 3 –

Update description to include new details (as described above). This is done for consistency. I think some of the existing text should be part of the subscription CREATE description and perhaps modifications are warranted. See comments in text for changes that I propose.

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

### 9.6.8 Resource Type *subscription*

The *<subscription>* resource contains subscription information for its subscribed-to resource.

A subscription to a resource allows an entity in the oneM2M architecture to be notified about changes of the subscribed-to resource. The *<subscription>* resource shall represent a subscription to a subscribed-to resource. In order to establish a subscription, a *<subscription>* resource shall be created as a child resource of the subscribed-to resource. The <*subscription*> child resource contains information about the exact scope of the subscription and targets to be notified. For example, a *<container>* resource having a *<subscription>* resource as a child resource (see clause 9.6.6) shall result in notification(s) of target(s) configured in the <*subscription*> child resource when changes to the parent <*container*> resource matching with notification event criteria described by the child <*subscription*> resource occur. A *<subscription>* resource shall be deleted when the parent subscribed-to resource is deleted.

In general, an Originator shall be able to create a resource of *<subscription>* resource type when the Originator has RETRIEVE privilege to the subscribed-to resource. The Originator which creates a *<subscription>* resource becomes the resource subscriber.

A <subscription> resource can be configured to implement a blocking “UPDATE” to a resource or attributes of a resource whereby a notification is sent to the notification target to respond with the result of the “UPDATE” request.

Each *<subscription>* may include notification policies that specify which, when, and how notifications are sent. These notification policies may work in conjunction with CMDH policies.

When a *<subscription>* resource is deleted, a Notify request shall be sent to the target indicated by the attribute *subscriberURI* if it is provided by the Subscriber.

The *<subscription>* resource shall contain the child resources specified in table 9.6.8-1.

Table 9.6.8-1: Child resources of *<subscription>* resource

| Child Resources of <*subscription*> | Child Resource Type | Multiplicity | Description |
| --- | --- | --- | --- |
| *notificationSchedule* | *<schedule>* | 0..1 | In the context of the *<subscription>* resource, the *notificationSchedule* specifies when notifications may be sent by the Hosting CSE to the *notificationURI(s).* See clause 9.6.9. |
| *[variable]* | *<notificationTargetMgmtPolicyRef>* | 0..n | See 9.6.31 for this type of resource.  |
| *nstr* | *<notificationTargetSelfReference>* | 1 | See 9.6.34 for this type of resource. |
| *[variable]* | *<transaction>* | 0..n | See clause 9.6.48 |

The *<subscription>* resource shall contain the attributes specified in table 9.6.8-2.

Table 9.6.8-2: Attributes of *<subscription>* resource

| Attributes of *<subscription>* | Multiplicity | RW/RO/WO | Description |
| --- | --- | --- | --- |
| *resourceType* | 1 | RO | See clause 9.6.1.3. |
| *resourceID* | 1 | RO | See clause 9.6.1.3. |
| *resourceName* | 1 | WO | See clause 9.6.1.3. |
| *parentID* | 1 | RO | See clause 9.6.1.3. |
| *expirationTime* | 1 | RW | See clause 9.6.1.3. |
| *creationTime* | 1 | RO | See clause 9.6.1.3. |
| *lastModifiedTime* | 1 | RO | See clause 9.6.1.3. |
| *labels* | 0..1 (L) | RW | See clause 9.6.1.3. |
| *accessControlPolicyIDs* | 0..1 (L) | RW | See clause 9.6.1.3.If no *accessControlPolicyIDs* value is configured, the *accesControlPolicyIDs* of the parent resource shall be applied for privilege checking. |
| *dynamicAuthorizationConsultationIDs* | 0..1 (L) | RW | See clause 9.6.1.3. |
| *creator* | 0..1 | WO | See clause 9.6.1.3. |
| *eventNotificationCriteria* | 0..1 | RW | This attribute (notification policy) indicates the event criteria for which a notification is to be generated. When no *eventNotificationCriteria* attribute is present in a <*subscription*> resource, the Hosting CSE shall trigger notifications for this subscription when any of the attributes of the subscribed-to resource is modified. |
| *expirationCounter* | 0..1 | RW | This attribute (notification policy) indicates that the subscriber wants to set the life of this subscription to a limit of a maximum number of notifications. When the number of notifications sent reaches the count of this counter, the *<subscription>* resource shall be deleted, regardless of any other policy. |
| *notificationURI* | 1 (L) | RW | This attribute shall be configured as a list consisting of one or more targets that the Hosting CSE shall send notifications to. A target shall be formatted as a oneM2M compliant Resource-ID as defined in clause 7.2 or as an identifier compliant with a oneM2M supported protocol binding (e.g. http, coap, mqtt). If a target is formatted as a oneM2M compliant Resource-ID, then the target shall be formatted as a structured or unstructured CSE-Relative-Resource-ID, SP-Relative-Resource-ID, and/or Absolute-Resource-ID of an <*AE*> or <CSEBase> resource. A Hosting CSE shall use this information to determine proper pointOfAccess, requestReqchability and/or pollingChannel information needed to send a notification to the target. The following is an example.* /CSE0001/AE0001

For a target that is formatted as an identifier compliant with a oneM2M supported protocol binding, the details of this format are defined by the respective oneM2M protocol specification. The following is an example of an HTTP URI compliant with oneM2M HTTP protocol binding.* <https://172.25.30.25:7000/notification/handler>

For a subscription to a <fanoutpoint> resource, if <subscription> resource in request contains a notificationForwardingURI, then the group hosting CSE shall configure the *notificationURI* of the fanout subscription request with an address specified by the Group Hosting CSE that can be used by the Group Hosting CSE to receive aggregated notifications. A notification serialization type may be appended to each notification target configured in this list. The Hosting CSE shall serialize notifications and send it to a notification target based on this serialization type indicator. Possible serialization types are defined in the TS-0004 [3] (e.g. XML, JSON or CBOR). If a notification serialization type is not appended to a notification target, a default shall apply based on the Hosting CSE local policy. The syntax for appending a serializatino type to a notification target shall use the “?” delimiter character as shown in the below examples.* <http://mydomain/notificationHandler?ct=json>
* CSE02/base/ae2?ct=xml
 |
| *groupID* | 0..1 | RW | The ID of a *<group>* resource in case the subscription is made through a group. This attribute may be used in the ***Filter Criteria*** to discover all subscription resources created via a <fanOutPoint> resource to a specific groupID. |
| *notificationForwardingURI* | 0..1(L) | RW | The attribute shall be present only for group related subscriptions. If the subscriber intends the Group Hosting CSE to aggregate the notifications, the attribute shall be set identical to the *notificationURI* attribute. It shall be used by Group Hosting CSE for forwarding aggregated notifications. See clauses 10.2.7.10 and 10.2.7.11. |
| *batchNotify* | 0..1 | RW | This attribute (notification policy) indicates that the subscription originator wants to receive batches of notifications rather than receiving them one at a time. This attribute includes: the number of notifications to be batched for delivery and the duration. When only the number is specified by the subscription originator, the Hosting CSE shall set the default duration given by M2M Service Provider. If *batchNotify* is used simultaneously with *latestNotify*, only the latest notification shall be sent and have the ***Event Category*** set to "latest". |
| *rateLimit* | 0..1 | RW | This attribute (notification policy) indicates that the subscriber wants to limit the rate at which it receives notifications. This attribute expresses the subscriber's notification policy and includes two values: a maximum number of events that may be sent within some duration, and the *rateLimit* window duration. When the number of generated notifications within the *rateLimit* window duration exceeds the maximum number, notification events are temporarily stored, until the end of the window duration, when the sending of notification events restarts in the next window duration. The sending of notification events continues as long as the maximum number of notification events is not exceeded during the window duration. The *rateLimit* policy may be used simultaneously with other notification policies. |
| *preSubscriptionNotify* | 0..1 | WO | This attribute (notification policy) indicates that the subscriber wants to be sent notifications for events that were generated prior to the creation of this subscription. This attribute has a value of the number of prior notification events requested. If up-to-date caching of retained events is supported on the Hosting CSE and contains the subscribed events, then prior notification events will be sent up to the number requested. The *preSubscriptionNotify* policy may be used simultaneously with any other notification policy. |
| *pendingNotification* | 0..1 | RW | This attribute (notification policy), if set, indicates how missed notifications due to a period of no connectivity are handled (according to the reachability and notification schedules). The possible values for *pendingNotification are*:* "sendLatest";
* "sendAllPending".

This policy depends upon caching of retained notifications on the hosted CSE. When this attribute is set to "sendLatest", only the last notification shall be sent and it shall have the ***Event Category*** set to "latest". If this attribute is not present, the Hosting CSE sends no missed notifications. This policy applies to all notifications regardless of the selected delivery policy (*batchNotify*, *latestNotify*, etc.) Note that unreachability due to reasons other than scheduling is not covered by this policy. |
| *notificationStoragePriority* | 0..1 | RW | Indicates that the subscriber wants to set a priority for this subscription relative to other subscriptions belonging to this same subscriber. This attribute sets a number within the priority range. When storage of notifications exceeds the allocated size, this policy is used as an input with the storage congestion policy (*notificationCongestionPolicy*) specified in clause 9.6.3 to determine which stored and generated notifications to drop and which ones to retain. |
| *latestNotify* | 0..1 | RW | This attribute (notification policy) indicates if the subscriber wants only the latest notification. If multiple notifications of this subscription are buffered, and if the value of this attribute is set to true, then only the last notification shall be sent and it shall have the ***Event Category*** value set to "latest". |
| *notificationContentType* | 1 | RW | Indicates a notification content type that shall be contained in notifications. The allowed values are:* "modified attributes";
* "all attributes";
* "ID" of the resource indicated in the *notificationEventType* condition.
* Trigger Payload

If it is not given by the Originator at the creation procedure, default is "all attributes".The value “Trigger Payload” for this attribute is only valid when at least one “*notificationEventType”* tag in the *eventNotificationCriteria* attribute contains the event “Trigger Received targeting the MN/ASN-AE associated with the <*AE*> parent resource”. |
| *notificationEventCat* | 0..1 | RW | This attribute (notification policy) indicates the subscriber's requested ***Event Category*** to be used for notification messages generated by this subscription. |
| *subscriberURI* | 0..1 | WO | This attribute shall be configured with the target of the subscriber. The target is used by the Hosting CSE to determine where to send a notification when the subscription is deleted. A target shall be formatted as a oneM2M compliant Resource-ID as defined in clause 7.2 or as an identifier compliant with one of the oneM2M supported protocol bindings (the detailed format of which are defined by each respective oneM2M protocol binding specification). |
| *associatedCrossResourceSub* | 0..1 | RW | This attribute lists *the identifier of <crossResourceSubscription>* resources where this *<subscription>* is involved in.  |

Table 9.6.8-3 describes the *eventNotificationCriteria* conditions.

Table 9.6.8-3: *eventNotificationCriteria* conditions

| Condition tag | Multiplicity | Matching condition |
| --- | --- | --- |
| *createdBefore* | 0..1 | The *creationTime* attribute of the resource is chronologically before the specified value. |
| *createdAfter* | 0..1 | The *creationTime* attribute of the resource is chronologically after the specified value. |
| *modifiedSince* | 0..1 | The *lastModifiedTime* attribute of the resource is chronologically after the specified value. |
| *unmodifiedSince* | 0..1 | The *lastModifiedTime* attribute of the resource is chronologically before the specified value. |
| *stateTagSmaller* | 0..1 | The *stateTag* attribute of the resource is smaller than the specified value. |
| *stateTagBigger* | 0..1 | The *stateTag* attribute of the resource is bigger than the specified value. |
| *expireBefore* | 0..1 | The *expirationTime* attribute of the resource is chronologically before the specified value. |
| *expireAfter* | 0..1 | The *expirationTime* attribute of the resource is chronologically after the specified value. |
| *sizeAbove* | 0..1 | The *contentSize* attribute of the *<contentInstance>* resource is equal to or greater than the specified value. |
| *sizeBelow* | 0..1 | The *contentSize* attribute of the *<contentInstance>* resource is smaller than the specified value. |
| *notificationEventType* | 0..6 | The type of event that shall trigger a notification. If multiple *notificationEventType* tags are present, a notification shall be triggered if any of the configured events occur. Note that not all permutations of event type are meaningful. Possible notification event type values are: 1. Update to attributes of the subscribed-to resource
2. Deletion of the subscribed-to resource,
3. Creation of a direct child of the subscribed-to resource,
4. Deletion of a direct child of the subscribed-to resource
5. An attempt to retrieve a <*contentInstance*> direct-child-resource of a subscribed-to <*container*> resource is performed while this <*contentInstance*> child resource is an obsolete resource or the reference used for retrieving this resource is not assigned. This retrieval is performed by a RETRIEVE request targeting the subscribed-to resource with the Result Content parameter set to either "child-resources" or "attributes+child-resources". This value for the *eventNotificationType* tag implies that the subscribed-to resource shall be an <*container*> resource. Otherwise this setting is not valid.
6. Trigger Received targeting the MN/ASN-AE associated with the <AE> parent resource. This implies that the subscribed-to resource shall be an <*AE*> resource instance. Otherwise this setting is not valid.
7. Update to attributes of thesubscribed-to resource with blocking of the triggering UPDATE operation. For this *eventNotificationType* value setting, only one single Notification Target shall be present in the *notificationURI* attribute – see *notificationURI* attribute definition. This value for the *eventNotificationType* tag shall not be combined with any other *eventNotificationType* tag value. This value for *notificationEventType* establishes a subscription that is triggered for the same events as for the value “Update to attributes of the subscribed-to resource”. However, upon occurrence of a triggering UPDATE operation that has been validated and results in an authorized UPDATE operation, the triggering UPDATE operation shall be blocked by the Hosting CSE until a notification request was sent out and a corresponding response message was received or a timeout happens. When the response status code of the notification response message indicates a successful notification reception in combination with a successful notification action taken by the Notification Target entity, the triggering UPDATE operation shall be completed with a successful update of the targeted attribute(s). If the notification response message indicates an unsuccessful notification reception or a successful notification reception with unsuccessful notification action by the targeted entity or times out, the blocked UPDATE operation shall be completed with no success and no change of the targeted attribute(s). For any subscribed-to resource there shall exist a maximum of one subscription with this setting of *notificationEventType*. All other notification policies shall not be allowed when this setting of *notificationEventType* is used. The *notificationContentType* shall be “modified attibutes”. When an UPDATE operation has been blocked due to triggering this type of notification, any other occurring UPDATE or DELETE requests to the same resource shall be handled only after the blocked UPDATE operation has been completed.

The other conditions in *eventNotificationCriteria* conditions apply within the scope of the selected *notificationEventType.*For example, if notificationEventType is "Creation of a direct child of the subscribed-to resource" then other *eventNotificationCriteria* conditions is applied to the direct child resources of the subscribed-to resource.If this condition is not specified, the default value is "Update to attributes of the subscribed-to resource".The notion of "obsolete resource" is defined in clause 9.6.1.3.2 (Common attributes). |
| *operationMonitor* | 0..n | The operations and/or the Originators accessing the subscribed-to resource matches with the specified value. It allows monitoring which operation and/or which Originator is attempting to the access subscribed-to resource regardless of whether the operation is performed. This feature is useful to detect AEs that send requests to a subscribed-to resource and that result in a successful or failure response. Possible arguments are operation(s) (e.g.: CREATE, RETRIEVE, UPDATE, DELETE, NOTIFY) and/or Originator identifier(s).If a set of Originator identifier(s) is included in this tag and no operations are listed, any operations initiated from any of the indicated Originator(s) shall trigger a notification. If a set of operation(s) is included in this tag and no Originator identifier, any of the listed operations shall trigger a notification.If both, a set of Originator identifiers and a set of operations are listed, then any of the listed operations initiated from any of the listed Originators shall trigger the notification.When the *notificationEventType* tag is present in the *eventNotificationCriteria,* the value of the *operationsMonitor* tag is ignored if present in the *eventNotificationCriteria* attribute. |
| *attribute* | 0..n | A list of attribute names of a subscribed-to-resource. This list is only applicable when *notificationEventType* has a value of "Update to attributes of the subscribed-to resource". or “Update to attributes of the subscribed-to resource with blocking of the triggering UPDATE operation”.If this list is present, then it is used to specify a subset of a subscribed-to resource's attributes for which updates shall result in a notification. If ANY attribute specified on this list is updated, then a notification shall be generated. If an attribute that is not specified in this list is updated, then a notification shall not be generated. If this list is not presented, then the default attribute list is the full set of a subscribed-to resource's attributes. If ANY attribute of a subscribed-to resource is updated, then a notification shall be generated. |
| *childResourceType* | 0.. 1 (L) | A list of resource types. This list is only applicable when *notificationEventType* has a value of "Creation of a direct child of the subscribed-to resource ".If this list is present, then it is used to specify a subset of resource type for direct child resource of which creation shall result in a notification. If ANY resource type specified on this list is created, then a notification shall be generated. If a resource type that is not specified in this list is created, then a notification shall not be generated. If this list is not present, then the default resource type list is the full set of a direct child resource.  |
| *missingData* | 0..1 | The *missingData* includes two values: a minimum specified missing number of the Time Series Data within the specified window duration, and the window duration. The condition only applies to subscribed-to resources of type *<timeSeries>*.The first detected missing data point starts the timer associated with the window duration. The window duration is restarted upon its expiry until such time as the entire subscription is terminated or not refreshed. More details about NOTIFICATIONS related to data reporting is found in section 10.2.39 |
| *filterOperation* | 0..1 | Indicates the logical operation (AND/OR) to be used for the condition tags *createdBefore, createdAfter, modifiedSince, unmodifiedSince, stateTagSmaller, stateTagBigger, expireBefore, expireAfter, sizeAbove, sizeBelow*. The default value is logical AND. |

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

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

No mixed AND/OR filter operation will be supported.

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

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

#### 10.2.10.2 Create *<subscription>*

This procedure shall be used to request the creation of a new *<subscription>* resource to instruct the Hosting CSE to send notifications to configured Subscriber(s) for modifications of a subscribed-to resource. The generic create procedure is described in clause 10.1.2.

Table 10.2.10.2-1: *<subscription>* CREATE

| *<subscription>* CREATE  |
| --- |
| Associated Reference Point | Mca, Mcc and Mcc' |
| Information in Request message | All parameters defined in table 8.1.2-3 apply with the specific details for:***Content*:** The resource content shall provide the information as defined in clause 9.6.8 |
| Processing at Originator before sending Request | According to clause 10.1.2 with the following additions:The Request shall address a subscribable resourceThe Request shall include a <*subscription>* resource representation withtheattribute *notificationURI*If the *notificationURI* attribute includes Notification Target(s) which is/are not targeting the Originator, the Originator should send the request as non-blocking request (see clauses 8.2.2 and 9.6.12) |
| Processing at Receiver | According to clause 10.1.2 with the followingWhich is also the Hosting CSE shall validate the followings:* Check if the subscribed-to resource, addressed in the ***To*** parameter in the Request, is a subscribable resource
* Check if the Originator has privileges for retrieving the subscribed-to resource
* In case a <*subscription*> resource representation is provided with a *notificationEventType* tag equal to “Update to attributes of the subscribed-to resource with blocking of the triggering UPDATE operation” in the *eventNotificationCriteria* attribute, check that no other subscriptions with this setting exist for the resource in the ***To*** parameter, check that only one entity is targeted by the *notificationURI* attribute and check that this entity has privileges for updating the subscribed-to resource.
* If an entity listed in thenotificationURI is not the Originator, the Hosting CSE may send a Notify request to that entity to verify this *<subscription>* creation request. If the Hosting CSE initiates the verification, it shall check if the verification result in the Notify response is successful or not. If any of the entities listed in the *notificationURI* attribute fails verification then the *<subscription>* create process fails

If any of the checks above fails, the Hosting CSE shall send an unsuccessful response to the Originator with corresponding error information. Otherwise, the Hosting CSE shall create the *<subscription>* resource and send a successful response to the Originator. Upon successful creation of a <*subscription*> resource, the Hosing CSE shall evaluate subsequent operations on the subscribed-to resource and trigger notifications in line with the notification policies provisioned in the created <*subscription*> resource. |
| Information in Response message | All parameters defined in table 8.1.3-1 apply with the specific details for:* ***Content*:** address of the created *<subscription>* resource, according to clause 10.1.2
 |
| Processing at Originator after receiving Response | According to clause 10.1.2 |
| Exceptions | According to clause 10.1.2 |

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

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

#### 10.2.10.7 Notification message handling procedure

When a Hosting CSE receives a *<subscription>* creation request which requires verification (see clause 10.2.10.2), the Hosting CSE may send a notification to perform subscription verification. In this case, the notification shall include the ID of the Originator of the <subscription> resource creation.

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.

When a subscription shall be established that sends notifications upon update of attributes of the subscribed-to resource while blocking the triggering UPDATE operation until the result of the notification is received, the value of the *eventNotificationType* tag in the *notificationEventCriteria* attribute shall be set to “Update to attributes of the subscribed-to resource with blocking of the triggering UPDATE operation”, see clause 9.6.8. For this *eventNotificationType* value setting, only one single Notification Target shall be present in the *notificationURI* attribute – see *notificationURI* attribute definition in clause 9.6.8. A subset of attributes of the subscribed-to resource that are triggering a notification when modified can be specified in the *attribute* tag of the *notificationEventCriteria* attribute. If the *attribute* tag is not present, all attributes of the subscribed-to resource will trigger a notification when modified. Upon occurrence of a triggering UPDATE operation that has been validated and results in an authorized UPDATE operation for any of the triggering attributes of the subscribed-to resource, the triggering UPDATE operation shall be blocked before modifying the targeted attributes by the Hosting CSE until a notification request was sent out and a corresponding response message was received or a timeout happens. While such an UPDATE request is pending, no other UPDATE or DELETE requests to the same resource instance shall be processed, i.e. if they occur while the UPDATE operation that triggered this type of subscription is blocked, they need to be delayed until the blocked UPDATE has been completed. When the response status code of the notification response message indicates a successful notification reception by the Notification Target in combination with a successful notification action taken by the Notification Target, the blocked UPDATE operation shall be completed with a successful update of the targeted attribute(s). If the notification response message indicates an unsuccessful notification request reception or a successful notification request reception with unsuccessful notification action by the Notification Target or when the reception of a response message times out, the blocked UPDATE operation shall be completed with no success and no change of the targeted attribute(s). There shall exist a maximum of only one subscription with this setting of *notificationEventType* for a given resource. All other notification policies shall not be allowed when this setting of *notificationEventType* is used.

Further details of Hosting CSE related notification policies follow:

The *expirationCounter* shall be decreased by one when the Hosting CSE successfully sends the notification request to Receiver(s). If the counter reaches zero, the corresponding subscription resource shall be deleted.

In the case an Originator wants to create batches of notifications rather than have the Hosting CSE send notifications one by one, it may set the *batchNotify* attribute to express its notification policy. The *batchNotify* attribute (notification policy) is based on two values, the number of notifications to be batched for delivery, and/or a duration. When the Hosting CSE generates a notification event it checks the *batchNotify* policy, if a duration value is specified then a timer is started which expires after the duration value. If a number of notifications is specified then notification events are accumulated until the accumulated notification events reaches the specified number. If only the duration is specified, then the accumulated notifications are sent as a batch when the timer expires. If both values are set then accumulated notifications are sent as a batch when either the timer expires or the number is reached whichever happens first. If neither the number nor the duration is specified (i.e. the *batchNotify* attribute is present and empty), then the Hosting CSE shall batch notifications using the default duration value as given by the M2M Service Provider. Note that Hosting CSE shall not batch notifications when the *batchNotify* is not present in the <subscription> resource. When the first notification event is generated then a timer shall be started and keep batching notifications for the duration. After the duration, batched notification shall be sent and a timer shall be set again at the next notification event. For example, a *batchNotify* policy having a duration of 10 minutes and a number of 20 notifications will accumulate notifications which is sent when the first of these two conditions are satisfied. The sending order is first-in first out (FIFO). The batch timer shall be reset once the batched notifications are being sent. *notificationEventCat* is checked at the time of batch transmission and applied to each notification individually in the batch. Stored notification events may be dropped according to the *notificationStoragePriority* and the *notificationCongestionPolicy* (see clause 9.6.3). When the *batchNotify* and *latestNotify* attributes (notification policies) are used together, they enable two ways of sampling notification events for notification generation. If the number of notification is set high then the duration value will drive the policy, and the *latestNotify* policy will cause a single event notification every duration period, e.g. send the latest event notification every hour. If the duration value is set high then the number of notifications will drive the policy, and the *latestNotify* policy will cause a single notification for every specified number of notifications, e.g. send the latest event notification for every 500 events notifications generated. The scope of the *batchNotify* policy is the Hosting CSE for the one subscription it is set in, and does not extend to transit CSEs.

In the case when an Originator wants to limits the rate at which notifications are sent, it may set the *rateLimit* attribute (notification policy) to express its notification policy. The *rateLimit* policy is based on two values, a maximum specified number of events (e.g. 10, 000) that may be sent within some specified *rateLimit* window duration (e.g. 60 seconds), and the *rateLimit* window duration. When the Hosting CSE generates a notification event it checks the *rateLimit* policy and whether the current total number of events sent is less than the maximum number of events within the current *rateLimit* window duration. If the current total is less than the maximum number then the notification may be sent. If it is equal or more then the notification is temporarily stored until the end of the current window duration, when the sending of notification events restarts in the next window duration. The sending of notification events continues as long as the maximum number of notification events is not exceeded within the window duration. The *rateLimit* windows are sequential (not rolling). The *rateLimit* policy may be used simultaneously with *batchNotify* and *notificationStoragePriority* policies. The scope of the *rateLimit* policy is the Hosting CSE for the one subscription it is set in, and does not extend to transit CSEs.

The *pendingNotification* attribute (notification policy) indicates the notification procedure to be followed following a connectionless period (due to lack of notification schedule or reachability schedule). When the Hosting CSE generates a notification with the *pendingNotification*, it shall check the notification schedule of the subscription and the reachability schedule associated with theNotification Target. If there is no restriction then the notification is immediately sent, otherwise the notification may be cached according to the *pendingNotification*. If caching of retained notifications is supported on the Hosting CSE and contains the subscribed events then pending notification (those that occurred during the connectionless period) will be sent to Notification Target per the *pendingNotification* policy. If it is set to the "sendLatest", most recent notification should be sent and it shall have the ***Event Category*** set to "latest". Figure 10.2.10.7‑1 illustrates an example for this case. If it is set to "sendAllPending", all the missed cached notifications should be sent in the order they occurred. Figure 10.2.10.7-2 illustrates an example of this case. The Hosting CSE may use the *pendingNotification* policy to determine whether and how many interim notifications to retain in its cache. The *pendingNotification* policy may be used simultaneously with any other notification policy, which would impact what would be sent during the connection period. The scope of the *pendingNotification* is the Hosting CSE for the one subscription it is set in, and does not extend to transit CSEs.

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

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.?