|  |  |
| --- | --- |
| CHANGE REQUEST | |
| Meeting ID:\* | SDS 42 |
| Source:\* | Neeta Meshram([neeta@cdot.in](mailto:neeta@cdot.in) ), Poornima Trivedi(poornima@cdot.in), Anupama Chopra([anupama@cdot.in](mailto:anupama@cdot.in)) C-DOT |
| Date:\* | 2019-09-23 |
| Reason for Change/s:\* | See the Introduction |
| CR against: Release | Release-4 |
| CR against: WI\* | Active <Work Item number>  MNT maintenance / < Work Item number(optional)>  Is this a mirror CR? Yes  No  mirror CR number: (Note to Rapporteur - use latest agreed revision)  STE Small Technical Enhancements / < Work Item number (optional)>  Only ONE of the above shall be ticked |
| CR against: TS/TR\* | TS-0001-Functional\_Architecture-V4\_2\_0 |
| Clauses \* | 10.2.7.10 |
| Type of change: \* | Editorial change  Bug Fix or Correction  Change to existing feature or functionality  New feature or functionality  Only ONE of the above shall be ticked |
| Other TS/TR(s) impacted |  |
| Post Freeze checking:\* | This CR contains only essential changes and corrections? YES  NO  This CR may break backwards compatibility with the last approved version of the TS? YES  NO |
| Template Version: January 2019 (do not modify) | |

**oneM2M Notice**

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

GUIDELINES for Change Requests:

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

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

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

Currently, group related subscription proposes for an originator to set nfURI identical to nURI to indicate aggregatedNotification. But this is an overhead for an application, an application can just include it with an empty value to indicate that it wants aggregation while CSE in turn can set its value to nURI on finding the presence of attribute in request.

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

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. |
| *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 included with an empty value.. 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   For a list of the default and allowed values of *notificationContentType* for each of the supported values of *notificationEventType* refer to Table 9.6.8-4. |
| *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. |

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

**-----------------------Start of change 2---------------------------------------------**

#### Subscribe and Un-Subscribe *<fanOutPoint>* of a group

This procedure shall be used for receiving information about modifications of all member resources belonging to an existing *<group>* resource.

Table 10.2.7.10-1: *<fanOutPoint>* Subscribe/Un-subscribe

|  |  |
| --- | --- |
| *<fanOutPoint>* Subscribe/Un-subscribe | |
| Associated Reference Point | Mca, Mcc and Mcc' |
| Information in Request message | ***From:*** Identifier of the AE or CSE that initiates the request  ***To:*** The address of the <fanOutPoint> resource appended with the ID of the *<subscription>* resource to be created  ***Group Request Identifier:*** The group request identifier |
| Processing at Originator before sending Request | The Originator shall request to create a subscription resource under all member resources belonging to an existing *<group>* resource by using a CREATE operation. The request may address the virtual child resource *<fanOutPoint>* of the specific *<group>* resource of a group Hosting CSE. The request may also address the address that results from appending a relative address to the *<fanOutPoint>* in order to create the corresponding subscription to the resource represented by the relative address with respect to all member resources. In both cases the targeted resource shall the parent of the newly created <subscription> resource(s). The request shall include *notificationForwardingURI* attribute if the Originator wants the group Hosting CSE to aggregate the notifications. The request shall include the required information and may include the optional information as described in subscription management clause 10.2.10. The Originator may be an AE or a CSE |
| Processing at Group Hosting CSE | The *<group>* Hosting CSE shall:   * Check if the Originator has CREATE privilege in the *<accessControlPolicy>* resource referenced by the *membersAccessControlPolicyIDs* in the group resource. In the case *membersAccessControlPolicyIDs* is not provided the access control policy defined for the group resource shall be used * If the subscription resource in the request contains an *notificationForwardingURI* attribute, assign a URI to replace the *notificationURI* of the subscription resource which will be used to receive notifications from member hosting CSEs. The ID of the *<group>* resource shall be set to the *groupID* attribute of the *<subscription>* resource. The group Hosting CSE shall maintain the mapping of the generated *notificationURI* and the former *notificationURI* * Upon successful validation, obtain the IDs of all member resources from the   attribute membersIDs of the addressed <group> resource   * Generate fan out requests addressing the obtained address (appended with   the relative address if any) to the member hosting CSEs as indicated in figure  10.2.7.1-1. From parameter in the request is set to ID of the Originator from  the request from the original Originator   * If the group Hosting CSE determines that multiple members resources belong to one CSE according to the IDs of the member resources, it may converge the requests accordingly before sending out. This may be accomplished by the *<group>* Hosting CSE creating a *<group>* resource on the members Hosting CSE to collect all the members on that members Hosting CSE * After receiving the responses from the member hosting CSEs, respond to the Originator with the aggregated results and the associated members list |
| Processing at Member Hosting CSE | For the subscribe/un-subscribe procedure, the Members Hosting CSE shall treat the request received from the group Hosting CSE as a normal SUBSCRIBE request on the addressed member resource as if it comes from the original Originator. Therefore the members Hosting CSE shall:   * Check if the request has a group request identifier. Check if the group request   identifier is contained in the requested identifier stored locally. If match is  found, ignore the current request and respond an error. If no match is found,  locally store the group request identifier until the expiration of the request expiration time or local policy   * Check if the original Originator has the READ permission on the members resource * Upon successful validation, perform the subscribe procedures for the corresponding type of member resource as described in clause 10.2.10 * Send the corresponding response to the group Hosting CSE |
| Information in Response message | Converged responses from member hosting CSEs |
| Processing at Originator after receiving Response | None |
| Exceptions | * Same request with identical request identifier received * Originator does not have the access control privilege to access the *<fanOutPoint>* resource |

Un-subscribing to the members of a <group> resource uses the “Delete <fanOutPoint>” procedure defined in 10.2.7.9.

A typical example of how the subscription is established is as follows. The Originator is creating subscription resource on Member-1 resource, Member-2 resource and Member-3 resource. Member-2 resource and Member-3 resource are members of Group-2 resource. Member-1 resource and Group-2 resource are members of Group-1 resource. In this case, Group-2 resource is the sub-group of Group-1 resource.

Originator

Group-1

Member-1

Group-2

Member-2

Member-3

*<subscription>*

*notificationURI* = address-1

*notificationForwardingURI =* address-1

*<subscription>*

*notificationURI* = address-2

*notificationForwardingURI =* address-1

*<subscription>*

*notificationURI* = address-3

*notificationForwardingURI =* address-1

Figure 10.2.7.10-1: Example of subscription through group

Originator sends the *<subscription>* resource creation request to *<fanOutPoint>* of Group-1 resource. The Originator intends the Group-1 Hosting CSE to aggregate the notifications, thus, the Originator includes the *notificationForwardingURI* along with *notificationURI* which is address-1 which is the address where the notification is supposed to be sent.

On receiving the request, the Group-1 Hosting CSE fans out the *<subscription>* creation request to address Member-1 resource and *<fanOutPoint>* resource of Group-2 resource. As *notificationForwardingURI* is set by the Originator, the Group-1 Hosting CSE allocates address-2 to receive aggregated notifications and put address-2 in the *notificationURI* of *<subscription>* resource to be fanned out.

On receiving the request, the Group-2 Hosting CSE fans out the *<subscription>* creation request to address Member-2 resource and Member-3 resource. As *notificationForwardingURI* is set, the Group-2 Hosting CSE allocates address-3 to receive aggregated notifications and put address-3 in the *notificationURI* of *<subscription>* resource to be fanned out. The mapping between address-2 and address-3 is maintained by the Group-2 Hosting CSE.

On receiving the request by any of the Member Hosting CSE, *<subscription>* resource is created.

**-----------------------End of change 2---------------------------------------------**

**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.?