|  |
| --- |
|  |

|  |  |
| --- | --- |
| CHANGE REQUEST | |
| Meeting ID:\* | SDS #42 |
| Source:\* | Convida Wireless |
| Date:\* | 2019-09-16 |
| Contact:\* | Dale Seed, Convida, [Seed.Dale@convidawireles.com](mailto:Seed.Dale@convidawireles.com)  Lu Liu, Convida, [Liu.Lu@convidawireless.com](mailto:Liu.Lu@convidawireless.com)  Catalina Mladin, Convida, [Mladin.Catalina@convidawireles.com](mailto:Mladin.Catalina@InterDigital.com) |
| Reason for Change/s:\* | Provides potential solution affecting process management for action triggering. |
| CR against: Release\* | Release 4 |
| CR against: WI\* | Active <WI-0093>  MNT / < Work Item number(optional)>  Is this a companion CR? Yes  No  Companion CR number: (Note to Rapporteur - use latest agreed revision)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\* | TR-0060 V 0.0.1 |
| Clauses/Sub Clauses \* | Clause 7 |
| 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) | N/A |
| 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

The following introduces the procedures for state and process management based on the resources introduced in SDS-2019-0222R01.



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

# Candidate Solutions

## Solution 1: State and Sequencing for Action Triggering

### 7.2.1 New Resource Type: *<state>*

The *<state>* resource is proposed to store the information of a state and define the actions and transitions in this state.

**Table 7.2.1-1: Attributes of *<state>* resource**

| **Attributes of  *<state>*** | **Multiplicity** | **RW/**  **RO/**  **WO** | **Description** |
| --- | --- | --- | --- |
| *currentStatus* | 1 | RO | The indicator of whether this state is currently active. This attribute may take values from “active” or “inactive”. |
| *stateActions* | 0..1 (L) | RW | The link(s) to the <action>(s) that may be performed in this state. The actions include both the ones that will be performed unconditionally when entering this state, and the ones that may be triggered by certain events/conditions within this state. |
| *stateTransitions* | 0..1 (L) | RW | The possible transition(s) that may happen from the current state. Each transition is defined as a tuple [transition criteria, next state]:  Transition criteria: the event or condition that may trigger state transition;  Next state: the *resourceID* of the next state to transition to.  If this state is the last state in the process, this attribute will be empty. |

After the IoT process enters a new state, the value of the *currentStatus* attribute will be changed to “active”. A timer will be started if the state has any duration constraint. Then, for all the <action>s that are linked through *stateActions* attribute, their *evalMode* attributes will be changed from “off” to “once/periodic/continuous”. For an unconditional action, it will be performed immediately; for a conditional action, the Hosting CSE will check or start to monitor the *evalCriteria* attribute of the <*action*> to determine whether the action will be performed or not. In addition, the transition criteria defined in *stateTransitions* will start to be monitored. If (one of) the event(s) defined in transition criteria happens or (one of) the defined condition(s) is met, the state transition will be triggered. At transition, the *currentStatus* will be changed to “inactive” and all the *evalMode* attributes of the corresponding actions will be changed back to “off”. If there is any ongoing action when the transition is triggered, the action will be completed or forced to end based off of local CSE policies. The Hosting CSE will then transition to the next state as indicated by the next state ID.

### 7.2.2 New Resource Type: <*processManagement>*

The *<processManagement>* resource is proposed to store the information of the entire process consisting of multiple states.

**Table 7.2.2-1: Child Resources of Proposed oneM2M <*processManagement*> Resource**

| **Child Resources of *<processManagement>*** | **Child Resource Type** | **Multiplicity** | **Description** |
| --- | --- | --- | --- |
| *[variable]* | *<state>* | 0..n | This resource describes the details of a particular state of an IoT process. |
| *[variable]* | *<subscription>* | 0..n | See clause 9.6.8 in [x]. |

**Table 7.2.2-2: Attributes of *<processManagement>* resource**

| **Attributes of  *<processManagement>*** | **Multiplicity** | **RW/**  **RO/**  **WO** | **Description** |
| --- | --- | --- | --- |
| *processStatus* | 1 | RO | The status for the entire IoT process. The supported values for this attribute are:  Disabled: The process is disabled.  Enabled: The process is enabled, and the Hosting CSE will start to monitor the event/condition defined in *preconditions*, if the pre-condition is met, the *processStatus* will become “Active”.  Active: The process is active, which means there is one state (other than *finalState*) currently active.  Paused: The process is paused and will remain in the current state and will not transition to another state until “Active” again.  Ended: The process has entered the final state or exited through *exitConditions*. |
| *processControl* | 1 | RW | This attribute is used to control and set the status of the process. The supported values for the attribute are:  Enable: to enable the process and update *processStatus* to “Enabled”.  Disable: to disable the process and update *processStatus* to “Disabled”.  Activate: to activate the process and update *processStatus* to “Active”.  Pause: to pause the process and update *processStatus* to “Paused”. |
| *currentState* | 1 | RO | The *resourceID* of the <*state*> resource that the process is currently in. If the *processStatus* is not “Active”, the value of this attribute will be empty. |
| *preconditions* | 0..1 (L) | RW | This attribute specifies any conditions that must be met for the process to begin. It can be used to trigger the start of the process. When the conditions are met, the *processStatus* will become “Active”, and the Hosting CSE will transition to the state defined in *initialState*. |
| *exitConditions* | 0..1 (L) | RW | This attribute specifies what events can end the IoT process. It allows for the premature and asynchronous exit of the process. When the exit conditions are met, the *processStatus* will become “Ended”, and all states in this process will be “inactive”. |
| *initialState* | 1 | RW | The *resourceID* of the first <*state*> resource of the process. In other words, this is the state transitioned to after “*preconditions*” are met. |
| *finalState* | 0..1 | RW | The *resourceID* of the last <*state*> resource in the process. If the process is a loop, this attribute is empty. |

### 7.2.3 State and Process Management

7.2.3.1 Introduction

This clause describes the procedure for managing oneM2M state and process in action triggering via the <*state*> and <*processManagement*> resources.

7.2.3.2 Process Management Procedure

This clause describes the procedure a Hosting CSE performs to manage an IoT process.

**Step 1:** An AEcreates a <*processManagement*> resource either as the child resource of the Hosting CSE’s <CSEBase> or elsewhere in the resource tree.

**Step 2:** An AEcreates all the <*action*> resources that are involved in the IoT process and state transitions.

**Step 3:** An AEcreates all the <*state*> resources that are involved in the IoT process as child resources of <*processManagement*>.

**Step 4:** An AEenables process management by updating the value of *processControl* to “Enable”. The Hosting CSE will in turn update the value of *processStatus* to “Enabled”. The Hosting CSE will start to monitor the event/condition defined in *preconditions*.

**Step 5:** If/when the event/condition defined in *preconditions* is detected, the Hosting CSE starts the process. The *processStatus* is set to “Active” by the Hosting. The Hosting CSE also transitions the process state to the state defined in *initialState*. If no preconditions are set, the process can be manually activated by setting the value of *processControl* to “Activate”.

**Step 4:** Starting from the *initialState*, whenever a state transition happens, the *currentState* attribute is updated to the *resourceID* of the new state by the Hosting CSE. Detailed procedures for state transitions are shown in the next clause.

**Step 5:** If *exitConditions* is defined, the Hosting CSE will start to monitor the event/condition defined in this attribute.

**Step 6:** If an AE updates *processControl* to “Pause”, and the Hosting CSE will pause the execution of the process and the *processStatus* will be updated to “Paused” accordingly. The paused process will remain in the current state and will not transition to another state until *processControl* is updated to “Activate”.

**Step 7:** An AE can also update *processControl* to “Disable” to disable the process and the Hosting CSE will disable execution of the process and the *processStatus* will be updated to “Disabled” accordingly.

**Step 8:** If the condition defined in *exitConditions* is met, the Hosting CSE will set *processStatus* to “Ended”, and all states in this process will be set to “inactive”.

**Step 9:** If the current state transitions to *finalState*, the *processStatus* will be set to “Ended” by the Hosting CSE.

7.2.3.3 State Transition Procedure

This clause describes the procedure a Hosting CSE performs to manage a state and the corresponding state transition.

**Step 1:** <*state*> resources are created as child resources of the <*processManagement*> resource, which corresponds to the IoT process that these states belong to.

**Step 2:** The IoT process enters a new state, e.g. after a state transition, or a process is initialized when the pre-conditions are met.

**Step 3:** The value of the *currentStatus* attribute is changed to “active”. If the state has any duration constraint, a timer will be started.

**Step 4:** When a state is activated, all the corresponding state specific actions (actions that could be performed in this state) will start to be monitored. As a result, for all the <*actions*> that are linked through the *stateActions* attribute, their *evalMode* attributes are changed from “off” to “once/periodic/continuous”.

For an unconditional action (performed unconditionally when entering this state), it will be performed immediately.

For a conditional action:

* if the *evalMode* attribute is “once”, the Hosting CSE will check the *evalCriteria* attribute of the <*action*> to determine whether the action will be triggered or not.
* if the *evalMode* attribute is “periodic” or “continuous”, the Hosting CSE will start to monitor the *evalCriteria* attribute of the <*action*> to determine if/when the action will be triggered.

**Step 5:** The Hosting CSE will start to monitor the transition criteria defined in *stateTransitions*.

**Step 6:** If one of the events or conditions defined in the transition criteria of *stateTransitions* is detected by the Hosting CSE , the Hosting CSE shall transition the state of the process.

**Step 7:** If/when the value of the *currentStatus* attribute transitions to “inactive” by the Hosting CSE, if there is any ongoing action, it will be either completed or forced to end, based off of local CSE policies.

**Step 8:** For all the state specific <*actions*> that are linked through the *stateActions* attribute, their *evalMode* attributes are changed to “off” by the Hosting CSE.

**Step 9:** The Hosting CSE transitions to the next state as indicated by the next state ID.

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

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.?