

## Clarification Request

**References:** [Specified Tests 23.1](#)  
[Test: 8.5.17.10 After FAULT-to-NORMAL, Re-Notification of OFFNORMAL](#)

**Date of BTL-WG Response:** 2023-09-14

**Background:** [Test: 8.5.17.10 After FAULT-to-NORMAL, Re-Notification of OFFNORMAL](#)

### 8.5.17.10 After FAULT-to-NORMAL, Re-Notification of OFFNORMAL (UnconfirmedEventNotifications)

Reason for Change: Fix the Status\_Flags in step 7. Added verify of pCurrentState after each transition. Add steps to wait pTimeDelay time delay before verifying pCurrentState when the state transitions from NORMAL to OFFNORMAL.

Purpose: To verify that objects go to the NORMAL state after leaving the FAULT state, then transition to OFFNORMAL if the condition still exists.

Test Concept: Select a fault detecting object O1 which is able to detect OFFNORMAL conditions. Make O1 transition to an OFFNORMAL state and then transition to FAULT. Remove the condition causing the FAULT and verify O1 transitions from FAULT to NORMAL, then verify that the object transitions from NORMAL to the original OFFNORMAL state.

Configuration Requirements: O1 is configured to detect and report unconfirmed events and faults. O1 is configured to have no fault conditions present, and Event\_State is NORMAL. The 'Issue Confirmed Notifications' parameter shall have a value of FALSE.

#### Test Steps:

1. VERIFY pCurrentReliability = NO\_FAULT\_DETECTED
2. VERIFY pCurrentState = NORMAL
3. MAKE(O1transition to an off normal state)
4. WAIT pTimeDelay
5. BEFORE Notification Fail Time
  - RECEIVE UnconfirmedEventNotification-Request
    - 'Process Identifier' = (any valid process identifier),
    - 'Initiating Device Identifier' = IUT,
    - 'Event Object Identifier' = O1,
    - 'Time Stamp' = (any valid time stamp),
    - 'Notification Class' = (the notification class configured for O1),
    - 'Priority' = (the value configured for the transition),
    - 'Event Type' = (ET1, any valid off normal event type),
    - 'Message Text' = (optional, any valid message text),
    - 'Notify Type' = ALARM | EVENT,
    - 'AckRequired' = TRUE | FALSE,
    - 'From State' = NORMAL,
    - 'To State' = OFFNORMAL,
    - 'Event Values' = (property-values appropriate for O1 )
6. VERIFY pCurrentState = OFFNORMAL
7. MAKE(O1 enter a fault state)
8. BEFORE Notification Fail Time
  - RECEIVE UnconfirmedEventNotification-Request
    - 'Process Identifier' = (any valid process identifier),
    - 'Initiating Device Identifier' = IUT,
    - 'Event Object Identifier' = O1,
    - 'Time Stamp' = (the current local time or sequence number),
    - 'Notification Class' = (the notification class configured for O1),
    - 'Priority' = (the value configured for the transition),

- |  |                  |  |
|--|------------------|--|
|  | 'Event Type' =   | CHANGE_OF_RELIABILITY,   |
|  | 'Message Text' = | (optional, any valid message text),  |
|  | 'Notify Type' =  | ALARM   EVENT,   |
|  | 'AckRequired' =  | TRUE   FALSE,  |
|  | 'From State' =   | OFFNORMAL,   |
|  | 'To State' =     | FAULT,   |
|  | 'Event Values' = | ( (R1 any valid BACnetReliability),<br>(?T, T, ?, ?),<br>(A list of valid values for properties required<br>to be reported for O1, and 0 or more other<br>properties of O1)) |
9. MAKE(O1 clear the fault condition)
10. BEFORE Notification Fail Time  
RECEIVE UnconfirmedEventNotification-Request
- |  |                                  |  |
|--|----------------------------------|--|
|  | 'Process Identifier' =           | (any valid process identifier),  |
|  | 'Initiating Device Identifier' = | IUT,   |
|  | 'Event Object Identifier' =      | O1,  |
|  | 'Time Stamp' =                   | (TS1: any valid time stamp),   |
|  | 'Notification Class' =           | (the notification class configured for O1),  |
|  | 'Priority' =                     | (the value configured for the transition),   |
|  | 'Event Type' =                   | CHANGE_OF_RELIABILITY,   |
|  | 'Message Text' =                 | (optional, any valid message text),  |
|  | 'Notify Type' =                  | ALARM   EVENT,   |
|  | 'AckRequired' =                  | TRUE   FALSE,  |
|  | 'From State' =                   | FAULT,   |
|  | 'To State' =                     | NORMAL,  |
|  | 'Event Values' =                 | ( NO_FAULT_DETECTED,<br>(F, F, ?, ?),<br>(A list of valid values for properties required<br>to be reported for O1, and 0 or more other<br>properties of O1)) |
11. VERIFY pCurrentReliability = NO\_FAULT\_DETECTED
12. VERIFY pCurrentState = NORMAL
13. WAIT until TS1 + pTimeDelay
14. BEFORE Notification Fail Time  
RECEIVE UnconfirmedEventNotification-Request
- |  |                                  |  |
|--|----------------------------------|--|
|  | 'Process Identifier' =           | (any valid process identifier),              |
|  | 'Initiating Device Identifier' = | IUT,   |
|  | 'Event Object Identifier' =      | O1,  |
|  | 'Time Stamp' =                   | (the current local time or sequence number), |
|  | 'Notification Class' =           | (the notification class configured for O1),  |
|  | 'Priority' =                     | (the value configured for the transition),   |
|  | 'Event Type' =                   | ET1,   |
|  | 'Message Text' =                 | (optional, any valid message text),          |
|  | 'Notify Type' =                  | ALARM   EVENT,                               |
|  | 'AckRequired' =                  | TRUE   FALSE,                                |
|  | 'From State' =                   | NORMAL,                                      |
|  | 'To State' =                     | OFFNORMAL,                                   |
|  | 'Event Values' =                 | (property-values appropriate for O1)         |
15. VERIFY pCurrentReliability = NO\_FAULT\_DETECTED
16. VERIFY pCurrentState = OFFNORMAL

**Problem:**

- 1) The two notifications (step 10 and step 14) have to be treated as one single event when pTimeDelay=0. In my test case, there were 62-microseconds (0.000062 seconds) between fault-to-normal and normal-to-offnormal notifications. The execution of steps 11 and 12 is not possible.
- 2) If pTimeDelay = 0 then TS1 in step 13 is in the past. Waiting until timestamp TS1 is impossible.

**Question:**

- 1) Should the test be adjusted to account for a  $pTimeDelay=0$  between the two notifications in steps 10 and 14?
- 2) Should Step 13 be adjusted to ensure that the timestamp cannot be in the past?

**Response:**

- 1) **Yes**
- 2) **No. The wait in step 13 should be a timer and not an absolute time. BTL Working Group will adjust the test step.**