Clarification Request

Request from: Craig Gemmill (craig.gemmill@tridium.com)

References: [Enter any references this request may reference. Some examples are: "BTL Implementers Guide-17", "BTL Checklist-3.0.final", "ASHRAE 135.1-2003", etc]

Stage: ☒Request, ☒Listed, ☒Analysis, ☒Resolved

Background / Proposed Solution:

The device in question uses an alarm lifecycle that maintains outstanding alarms until they have returned to normal and been acknowledged. The to-normal transition is associated with the original to-offnormal transition and must be matched to the original transition.

If an event-generating object has an Event_Enable property with the to-offnormal bit set, it must also have the to-normal bit set, so that the to-normal transition can be used to clear the alarm. Similarly, if the Event_Enable property has the to-normal bit set, the to-offnormal bit must be set to enable the original to-offnormal transition so that the to-normal transition has an event record to clear.

The server device needs to maintain the collection of alarms in a state that is self-consistent according to its own rules in addition to the rules of BACnet. Therefore it must constrain the allowed values of the Event_Enable property. This is already done in the vendor specific configuration of the device. It must also be done for the BACnet configuration of the device.

The proposed solution for this device is to disallow writes to the Event_Enable property that have the to-offnormal bit cleared and the to-normal bit set, or vice versa. The expected return for this type of write would be an error with an Error Class of PROPERTY and an Error Code of VALUE_OUT_OF_RANGE, although a different error may be more appropriate.

This is similar to the behavior of a MultiState object that has 5 states, but only 3 of those states can be written. An attempt to write to the Present_Value with a disallowed state will return an Error-PDU.

Specific Test in question is 135.1-7.3.1.10

Question:

Is this acceptable behavior?

Response:

No. The BTL-WG believes that if the Event_Enable property is writable all values should be accepted. One solution is to make the Event_Enable property read-only.

The following test change is made to accommodate an Event_Enable property that is read-only.

7.3.1.10 Event_Enable Tests

Reason For Change: The test does not call out what to do if the Event_Enable property is read-only and the IUT cannot be configured as specified in the Configuration Requirements. This test also contains an

error in step 11. There should not be a wait after the event-triggering property is put into a FAULT state. There is no SSPC proposal for this test.

Dependencies: ConfirmedEventNotification Service Initiation Tests, 8.4; UnconfirmedEventNotification Service Initiation Tests, 8.5; ReadProperty Service Execution Tests, 9.18; WriteProperty Service Execution Tests, 9.22.

BACnet Reference Clauses: 12.1.23, 12.2.24, 12.3.20, 12.5.22, 12.6.26, 12.7.24, 12.11.10, 12.14.18, 12.15.18, 12.16.33, 12.17.17, 12.18.18, 12.19.18 and 12.23.23.

Purpose: To verify that notification messages are transmitted only if the bit in Event_Enable corresponding to the event transition has a value of TRUE. This test applies to Event Enrollment objects and Analog Input, Analog Output, Analog Value, Binary Input, Binary Output, Binary Value, Life Safety Point, Life Safety Zone, Loop, Multi-state Input, Multi-state Output, and Multi-state Value objects that support intrinsic reporting.

Test Concept: The IUT is configured such that the Event_Enable property indicates that some event transitions are to trigger an event notification and some are not. Each event transition is triggered and the IUT is monitored to verify that notification messages are transmitted only for those transitions for which the Event_Enable property has a value of TRUE.

Configuration Requirements: The Event_Enable property shall be configured with a value of TRUE for either the TO-OFFNORMAL transition or the TO-NORMAL transition and the other event transition shall have a value of FALSE. *If the Event_Enable property is read-only, follow the test steps as written and verify correct behavior for the value of the Event_Enable property.* For analog objects the Limit_Enable property shall be configured with the value (TRUE, TRUE). The referenced event-triggering property shall be set to a value that results in a NORMAL condition. If a Notification Class object is being used to configure recipient information the value of the Transitions parameter for all recipients shall be (TRUE, TRUE, TRUE).

In the test description below, "X" is used to designate the event-triggering property.

1. VERIFY Event_State = NORMAL
2. WAIT (Time_Delay + **Notification Fail Time**)
3. IF (X is writable) THEN
   WRITE X = (a value that is OFFNORMAL)
   ELSE
   MAKE (X have a value that is OFFNORMAL)
4. WAIT (Time_Delay)
5. BEFORE **Notification Fail Time**
   IF (the Transitions bit corresponding to the TO-OFFNORMAL transition is TRUE) THEN
   RECEIVE ConfirmedEventNotification-Request,
   'Process Identifier' = (any valid process ID),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (the event-generating object configured for this test),
   'Time Stamp' = (the current local time),
   'Notification Class' = (the class corresponding to the object being tested),
   'Priority' = (the value configured to correspond to a TO-OFFNORMAL transition),
   'Event Type' = (any valid event type),
   'Notify Type' = EVENT | ALARM,
   'AckRequired' = TRUE | FALSE,
   'From State' = NORMAL,
   'To State' = OFFNORMAL,
   'Event Values' = (values appropriate to the event type)
   ELSE

CHECK (verify that the IUT did not transmit an event notification message)
6. VERIFY Event_State = OFFNORMAL
7. IF (X is writable) THEN
   WRITE X = (a value that is NORMAL)
   ELSE
   MAKE (X have a value that is NORMAL)
8. WAIT (Time_Delay)
9. BEFORE **Notification Fail Time**
   IF (the Transitions bit corresponding to the TO-NORMAL transition is TRUE) THEN
   RECEIVE ConfirmedEventNotification-Request,
   'Process Identifier' = (any valid process ID),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (the event-generating object configured for this test),
   'Time Stamp' = (the current local time),
   'Notification Class' = (the class corresponding to the object being tested),
   'Priority' = (the value configured to correspond to a TO-NORMAL transition),
   'Event Type' = (any valid event type),
   'Notify Type' = EVENT | ALARM,
   'AckRequired' = TRUE | FALSE,
   'From State' = OFFNORMAL,
   'To State' = NORMAL,
   'Event Values' = (values appropriate to the event type)
   ELSE
   CHECK (verify that the IUT did not transmit an event notification message)
10. VERIFY Event_State = NORMAL
11. IF (the event-triggering object can be placed into a fault condition) THEN {
    MAKE (the event-triggering object change to a fault condition)
    ~~WAIT (Time_Delay)~~
    BEFORE **Notification Fail Time**
    IF (the Transitions bit corresponding to the TO-FAULT transition is TRUE) THEN

    RECEIVE ConfirmedEventNotification-Request,
    'Process Identifier' = (any valid process ID),
    'Initiating Device Identifier' = IUT,
    'Event Object Identifier' = (the event-generating object configured for this test),
    'Time Stamp' = (the current local time),
    'Notification Class' = (the class corresponding to the object being tested),
    'Priority' = (the value configured to correspond to a TO-FAULT transition),
    'Event Type' = (any valid event type),
    'Notify Type' = EVENT | ALARM,
    'AckRequired' = TRUE | FALSE,
    'From State' = NORMAL,
    'To State' = FAULT,
    'Event Values' = (values appropriate to the event type)
    ELSE
    CHECK (verify that the IUT did not transmit an event notification message)
    VERIFY Event_State = FAULT
    }

Notes to Tester: The UnconfirmedEventNotification service may be substituted for the ConfirmedEventNotification service. The 'Message Text' parameter is omitted in the test description because it is optional. The IUT may include this parameter in the notification messages.