

Clarification Request

References: 135.1-2013/BTL test 7.3.1.13

Date of BTL-WG Response: March 24, 2016

☒ All actions necessitated have been completed

Background:

Test 7.3.1.13 has known technical inaccuracies, since clauses 13.3.5 through 13.3.16 now allow optional transition directly, as:

If pCurrentState is HIGH_LIMIT, and pMonitoredValue is less than (pSetpoint - pLowDiffLimit) for pTimeDelay, then indicate a transition to the LOW_LIMIT event state.

If pCurrentState is LOW_LIMIT, and pMonitoredValue is greater than (pSetpoint + pHighDiffLimit) for pTimeDelay, then indicate a transition to the HIGH_LIMIT event state.

If pCurrentState is HIGH_LIMIT, and the LowLimitEnable flag of pLimitEnable is TRUE, and pMonitoredValue is less than pLowLimit for pTimeDelay, then indicate a transition to the LOW_LIMIT event state.

If pCurrentState is LOW_LIMIT, and the HighLimitEnable flag of pLimitEnable is TRUE, and pMonitoredValue is greater than pHighLimit for pTimeDelay, then indicate a transition to the HIGH_LIMIT event state.

7.3.1.13 Limit_Enable Test

Reason for Change vs version in 135.1-2013: Added a missing step to check that a notification is not sent.

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: 12.1.22, 12.2.23, and 12.3.19.

Purpose: To verify that the Limit_Enable property correctly enables or disables reporting of out of range events. This test applies to objects with a Limit_Enable property.

Test Concept: The event-triggering property is manipulated to cause both the high limit and the low limit to be exceeded for each possible combination of values for Limit_Enable. The resulting event notification messages are monitored to verify that they are transmitted only for circumstances where the associated event limit is enabled.

Configuration Requirements: Configure the object with High_Limit, Low_Limit and Deadband values such that High_Limit - Deadband > Low_Limit and both the Low_Limit and High_Limit values are within the valid range of values for Present_Value. If the device cannot be configured with limit values that meet these conditions, then this test shall be skipped. The Event_Enable property should be set to (TRUE, ?, TRUE) for this test. If the Event_Enable cannot be configured such that the TO-NORMAL and the TO-OFFNORMAL transitions are TRUE, this test may be skipped.

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

Test Steps:

1. IF Limit_Enable can be made to be equal (TRUE, TRUE)
2. If Limit_Enable is writable
 WRITE Limit_Enable = (TRUE, TRUE)
 ELSE
 MAKE Limit_Enable = (TRUE, TRUE)
3. WAIT (Time_Delay + **Notification Fail Time**)

```

4.      VERIFY Event_State = NORMAL
5.      IF (X is writable) THEN
          WRITE X = (a value that exceeds High_Limit)
        ELSE
          MAKE (X a value that exceeds High_Limit)
6.      WAIT (Time_Delay)
7.      BEFORE Notification Fail Time
          RECEIVE ConfirmedEventNotification-Request,
              'Process Identifier' =      (any valid process ID),
              'Initiating Device Identifier' = IUT,
              'Event Object Identifier' = (the 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' =              OUT_OF_RANGE,
              'Notify Type' =             ALARM | EVENT,
              'AckRequired' =             TRUE | FALSE,
              'From State' =              NORMAL,
              'To State' =                HIGH_LIMIT,
              'Event Values' =            (values appropriate to the event type)
8.      TRANSMIT SimpleAck-PDU
9.      IF (X is writable) THEN
          WRITE X = (a value that is lower than Low_Limit)
        ELSE
          MAKE (X a value that is lower than Low_Limit)
10     .  WAIT (Time_Delay)
11.      BEFORE Notification Fail Time
          RECEIVE ConfirmedEventNotification-Request,
              'Process Identifier' =      (any valid process ID),
              'Initiating Device Identifier' = IUT,
              'Event Object Identifier' = (the 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' =              OUT_OF_RANGE,
              'Notify Type' =             ALARM | EVENT,
              'AckRequired' =             TRUE | FALSE,
              'From State' =              HIGH_LIMIT,
              'To State' =                NORMAL,
              'Event Values' =            (values appropriate to the event type)
12.      TRANSMIT SimpleAck-PDU
13.      WAIT (Time_Delay)
14.      BEFORE Notification Fail Time
          RECEIVE ConfirmedEventNotification-Request,
              'Process Identifier' =      (any valid process ID),
              'Initiating Device Identifier' = IUT,
              'Event Object Identifier' = (the 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' = OUT_OF_RANGE,
        'Notify Type' = ALARM | EVENT,
        'AckRequired' = TRUE | FALSE,
        'From State' = NORMAL,
        'To State' = LOW_LIMIT,
        'Event Values' = (values appropriate to the event type)
15. TRANSMIT SimpleAck-PDU
16. IF (X is writable) THEN
        WRITE X = (a value that is between Low_Limit + deadband and High_Limit)
    ELSE
        MAKE (X a value that is between than Low_Limit + deadband and High_Limit)
17. WAIT (Time_Delay)
18. BEFORE Notification Fail Time RECEIVE ConfirmedEventNotification-Request,
        'Process Identifier' = (any valid process ID),
        'Initiating Device Identifier' = IUT,
        'Event Object Identifier' = (the 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' = OUT_OF_RANGE,
        'Notify Type' = ALARM | EVENT,
        'AckRequired' = TRUE | FALSE,
        'From State' = LOW_LIMIT,
        'To State' = NORMAL,
        'Event Values' = (values appropriate to the event type)
19. TRANSMIT SimpleAck-PDU
20. IF Limit_Enable can be made to equal (FALSE, TRUE)
21. IF Limit_Enable is writable
        WRITE Limit_Enable = (FALSE, TRUE)
    ELSE
        MAKE (Limit_Enable = (FALSE,TRUE))
22. IF (X is writable) THEN
        WRITE X = (a value that exceeds High_Limit)
    ELSE
        MAKE (X a value that exceeds High_Limit)
23. WAIT (Time_Delay)
24. BEFORE Notification Fail Time RECEIVE ConfirmedEventNotification-Request,
        'Process Identifier' = (any valid process ID),
        'Initiating Device Identifier' = IUT,
        'Event Object Identifier' = (the 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' = OUT_OF_RANGE,
        'Notify Type' = ALARM | EVENT,
        'AckRequired' = TRUE | FALSE,
        'From State' = NORMAL,
        'To State' = HIGH_LIMIT,
        'Event Values' = (values appropriate to the event type)
25. TRANSMIT SimpleAck-PDU
26. IF (X is writable) THEN
        WRITE X = (a value that is between Low_Limit and High_Limit-Deadband)
    ELSE
        MAKE (X a value that is between Low_Limit and High_Limit-Deadband)

```

27. WAIT (Time_Delay)

28. BEFORE **Notification Fail Time** RECEIVE ConfirmedEventNotification-Request,
 'Process Identifier' = (any valid process ID),
 'Initiating Device Identifier' = IUT,
 'Event Object Identifier' = (the 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' = OUT_OF_RANGE,
 'Notify Type' = ALARM | EVENT,
 'AckRequired' = TRUE | FALSE,
 'From State' = HIGH_LIMIT,
 'To State' = NORMAL,
 'Event Values' = (values appropriate to the event type)

29. TRANSMIT SimpleAck-PDU

30. IF (X is writable) THEN
 WRITE X = (a value that is lower than Low_Limit)
 ELSE
 MAKE (X a value that is lower than Low_Limit)

31. WAIT (Time_Delay + **Notification Fail Time**)

32. CHECK (verify that no notification message was transmitted)

33. IF (X is writable) THEN
 WRITE X = (a value that is between Low_Limit+Deadband and High_Limit)
 ELSE
 MAKE (X a value that is between Low_Limit+Deadband and High_Limit)

34. WAIT (Time_Delay + **Notification Fail Time**)

35. *CHECK (verify that no notification message was transmitted)*

36. IF Limit_Enable can be made to equal (TRUE, FALSE)

37. IF Limit_Enable is writable
 WRITE Limit_Enable = (TRUE, FALSE)
 ELSE
 MAKE (Limit_Enable = (TRUE, FALSE))

38. IF (X is writable) THEN
 WRITE X = (a value that exceeds High_Limit)
 ELSE
 MAKE (X a value that exceeds High_Limit)

39. WAIT (Time_Delay + **Notification Fail Time**)

40. CHECK (verify that no notification message was transmitted)

41. IF (X is writable) THEN
 WRITE X = (a value that is lower than Low_Limit)
 ELSE
 MAKE (X a value that is lower than Low_Limit)

42. WAIT (Time_Delay)

43. BEFORE **Notification Fail Time** RECEIVE ConfirmedEventNotification-Request,
 'Process Identifier' = (any valid process ID),
 'Initiating Device Identifier' = IUT,
 'Event Object Identifier' = (the 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' = OUT_OF_RANGE,
 'Notify Type' = ALARM | EVENT,
 'AckRequired' = TRUE | FALSE,
 'From State' = NORMAL,

```

        'To State' = LOW_LIMIT,
        'Event Values' = (values appropriate to the event type)
44. TRANSMIT SimpleAck-PDU
45. IF (X is writable) THEN
        WRITE X = (a value that is between Low_Limit + Deadband and High_Limit)
    ELSE
        MAKE (X a value that is between Low_Limit + Deadband and High_Limit)
46. WAIT (Time_Delay)
47. BEFORE Notification Fail Time RECEIVE ConfirmedEventNotification-Request,
        'Process Identifier' = (any valid process ID),
        'Initiating Device Identifier' = IUT,
        'Event Object Identifier' = (the 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' = OUT_OF_RANGE,
        'Notify Type' = ALARM | EVENT,
        'AckRequired' = TRUE | FALSE,
        'From State' = LOW_LIMIT,
        'To State' = NORMAL,
        'Event Values' = (values appropriate to the event type)
48. IF Limit_Enable can be made to equal (FALSE, FALSE)
49. IF Limit_Enable is writable
        WRITE Limit_Enable = (FALSE, FALSE)
    ELSE
        MAKE (Limit_Enable = (FALSE, FALSE))
50. IF (X is writable) THEN
        WRITE X = (a value that exceeds High_Limit)
    ELSE
        MAKE (X a value that exceeds High_Limit)
51. WAIT (Time_Delay + Notification Fail Time)
52. CHECK (verify that no notification message was transmitted)
53. IF (X is writable) THEN
        WRITE X = (a value that is lower than Low_Limit)
    ELSE
        MAKE (X a value that is lower than Low_Limit)
54. WAIT (Time_Delay + Notification Fail Time)
55. CHECK (verify that no notification message was transmitted)
56. IF (X is writable) THEN
        WRITE X = (a value that is between Low_Limit and High_Limit)
    ELSE
        MAKE (X a value that is between Low_Limit and High_Limit)

57. WAIT (Time_Delay + Notification Fail Time)
58. CHECK (verify that no notification message was transmitted)

```

Notes to Tester: The UnconfirmedEventNotification service may be substituted for the ConfirmedEventNotification service in which case the TD shall skip all of the steps in which a SimpleACK-PDU is sent. The 'Message Text' parameter is omitted in the test description because it is optional. The IUT may include this parameter in the notification messages.

Question:

Is it acceptable that this test be performed other than as shown at steps 11 and 14? Or if an implementation is compliant with 135-2012 but not with the test, should it be Skipped?

Response:

Yes, there may be a transition observed directly from HIGH_LIMIT to LOW_LIMIT in place of steps 11 through 14. The test purpose is still valid, so the test shall not be Skipped.