Clarification Request

References: "ASHRAE 135-2008", "ASHRAE 135.1-2009"

Background / Proposed Solution:

In ASHRAE 135 – 2008 the “in-alarm” bit of Status Flags depends on the Event-State Property.

For Test: 7.3.1.13 – Limit_Enable in ASHRAE 135.1 – 2009, in **Step 20 through 24** we set Limit_Enable property to [False, True] and thus when we change the PV to Low_Limit as in **Step 15** the Test checks for only that no event notification was transmitted in **Step 34.** Before this Step there should be a Step for **‘Check Event_State = NORMAL’.** Similarly wherever there is a Step for checking that no event-notification was sent, this Step **‘Check Event_State = NORMAL’** should also be included. This would enhance the testing scope and help developers to validate this functionality correctly.

7.3.1.13 Limit_Enable Test

Reason for Change: This test has been modified to allow for portions of it to be skipped if the Limit_Enable property is not modifiable, to always transition through NORMAL, and to not disable a limit when the object is in an OFFNORMAL state. This test is included in the SSPC proposal MSO-001.

Purpose: To verify that the Limit_Enable property correctly enables or disables reporting of out of range events. This test applies to Analog Input, Analog Output, and Analog Value objects that support intrinsic reporting.

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
22. WRITE Limit_Enable = (FALSE, TRUE)
23. ELSE
24. MAKE (Limit_Enable = (FALSE,TRUE))
25. IF (X is writable) THEN
WRITE X = (a value that exceeds High_Limit)
ELSE
MAKE (X a value that exceeds High_Limit)
26. WAIT (Time_Delay)
27. 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)
28. 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)
29. WAIT (Time_Delay)
30. BEFORE Notification Fail Time RECEIVE ConfirmdEventNotification-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)
31. TRANSMIT SimpleAck-PDU
32. 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)
33. WAIT (Time_Delay + Notification Fail Time)
34. CHECK (verify that no notification message was transmitted)
35. 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)
36. WAIT (Time_Delay + Notification Fail Time)
37. CHECK (verify that no notification message was transmitted)
38. IF Limit_Enable can be made to equal (TRUE, FALSE)
39. IF Limit_Enable is writable
WRITE Limit_Enable = (TRUE, FALSE)
ELSE
MAKE (Limit_Enable = (TRUE, FALSE))
40. IF (X is writable) THEN
WRITE X = (a value that exceeds High_Limit)
ELSE
MAKE (X a value that exceeds High_Limit)
41. WAIT (Time_Delay + Notification Fail Time)
42. CHECK (verify that no notification message was transmitted)
43. 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)
44. WAIT (Time_Delay)
45. 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)

46. TRANSMIT SimpleAck-PDU
47. 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)
48. WAIT (Time_Delay)
49. 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)
50. IF Limit_Enable can be made to equal (FALSE, FALSE)
51. IF Limit_Enable is writable
WRITE Limit_Enable = (FALSE, FALSE)
ELSE
MAKE (Limit_Enable = (FALSE, FALSE))
52. IF (X is writable) THEN
WRITE X = (a value that exceeds High_Limit)
ELSE
MAKE (X a value that exceeds High_Limit)
53. WAIT (Time_Delay + Notification Fail Time)
54. CHECK (verify that no notification message was transmitted)
55. 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)
56. WAIT (Time_Delay + Notification Fail Time)
57. CHECK (verify that no notification message was transmitted)
58. 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)

59. WAIT (Time_Delay + Notification Fail Time)
60. 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:

Can this step **'Check Event_State = NORMAL'** be included in Test 7.3.1.13, along with the Step for checking no event-notification was sent?

Currently Test 7.3.1.13 does not verify Event_State for all the combinations of Limit_Enable property.

Response:

No, prior to Protocol_Revision 13 the standard is unclear whether the event-state remains normal, or becomes one of the offnormal states. The BTL would prefer implementations treat Limit_Enable in the manner clarified in changes in addenda 135-2010af (Protocol_Revision 13).