BACnet® TESTING LABORATORIES
ADDENDA

Addendum misc2 to BTL Test Package 16.1

Revision 2
Revised 8/31/2020

Approved by the BTL Working Group on July 23, 2020.
Approved by the BTL Working Group Voting Members on August 24, 2020.
Published on September 2, 2020.

[This foreword and the "Overview" on the following pages are not part of this Test Package. They are merely informative and do not contain requirements necessary for conformance to the Test Package.]

FOREWORD

The purpose of this addendum is to present current changes being made to the BTL Test Package. These modifications are the result of change proposals made pursuant to the continuous maintenance procedures and of deliberations within the BTL-WG Committee. The changes are summarized below.

In the following document, language to be added to existing clauses within the BTL Test Package 16.1 is indicated through the use of *italics*, while deletions are indicated by ~~strikethrough~~. Where entirely new subclauses are proposed to be added, plain type is used throughout

In contrast, changes to BTL Specified Tests also contain a yellow highlight to indicate the changes made by this addendum. When this addendum is applied, all highlighting will be removed. Change markings on tests will remain to indicate the difference between the new test and an existing 135.1 test. If a test being modified has never existed in 135.1, the applied result should not contain any change markings. When this is the case, square brackets will be used to describe the changes required for this test.

Each addendum can stand independently unless specifically noted via dependency within the addendum. If multiple addenda change the same test or section, each future released addendum that changes the same test or section will note in square brackets whether or not those changes are reflected.

BTL-16.1misc2-1: Update Test Package to Reference 135.1-2019 - BTLWG-739

Overview:

The BTL Test Plan and BTL Specified Tests documents need to be updated to reference 135.1-2019.

- Same tests will now be deleted from BTL Specified Tests and the BTL Test Plan will be updated to reference 135.1-2019.
- Change tracking is updated in BTL Specified Tests to derive from 135.1-2019.
- Original tests in BTL Specified Tests that now exist in 135.1-2019 that were renumbered will be renumbered in BTL Specified Tests to match and the BTL Test Plan document is also updated to reference the new test numbers.

Changes:

Test Plan Changes

Change table 1.1
to include 135.1-2019 and remove 135.1-2013 and 135.1-2013q and 135.1-2013r
Remove all references to 135-2012 in this table and replace with single references to 135-2016

Change all references of 135.1-2013 to 135.1-2019.
Change all references of 135.1-2013q to 135.1-2019.

Note: The EPICS CONSISTENCY TESTS were changed by Addenda Fix1
~~Change BTL - 5 to 135.1-2019 - 5~~
Keep EPICS Section due to addenda 16.1 fix1

Note: This test was changed in Addenda Fix1
Change BTL - 7.1.X3 to BTL - 7.1.3.

Change test name of 7.3.1.10.1 to "Event_Enable Test for TO_OFFNORMAL and TO_NORMAL, and TO_FAULT"

Change BTL - 7.3.1.10.2 to 135.1-2019 - 7.3.1.10.2
Change BTL - 7.3.1.11 to 135.1-2019 - 7.3.1.11

Change BTL - 7.3.1.13.X1 to BTL - 7.3.1.13.1
Change BTL - 7.3.1.13.X2 to BTL - 7.3.1.13.2

Change 135.1-2013q - 7.3.1.X1.1 to 135.1-2019 - 7.3.1.31.1
Change 135.1-2013q - 7.3.1.X1.2 to 135.1-2019 - 7.3.1.31.2

Change BTL - 7.3.1.X4 to 135.1-2019 - 7.3.1.17
Change BTL - 7.3.1.X5 to 135.1-2019 - 7.3.1.18
Change BTL - 7.3.1.X6.1 to 135.1-2019 - 7.3.1.19.1
Change BTL - 7.3.1.X6.2 to 135.1-2019 - 7.3.1.19.2
Change BTL - 7.3.1.X6.3 to 135.1-2019 - 7.3.1.19.3

Note: These were modified in Fix1
Change BTL - 7.3.1.X7.1 to BTL - 7.3.1.20.1
Change BTL - 7.3.1.X7.2 to BTL - 7.3.1.20.2

Change BTL - 7.3.1.X8.1 to 135.1-2019 - 7.3.1.21.1
Change BTL - 7.3.1.X8.2 to 135.1-2019 - 7.3.1.21.2
Change BTL - 7.3.1.X9.1 to 135.1-2019 - 7.3.1.22.1
Change BTL - 7.3.1.X9.2 to 135.1-2019 - 7.3.1.22.2

Change BTL - 7.3.2.9.7 to 135.1-2019 - 7.3.2.9.7

Change BTL - 7.3.1.X20 - Non-zero writable Strike Count Test to BTL - 7.3.1.X20.1
Change BTL - 7.3.2.X41.Y10 - Strike Count Test to BTL - 7.3.2.X20.2

Change BTL - 7.3.2.21.3.1 to 135.1-2019 - 7.3.2.21.3.1
Change BTL - 7.3.2.21.3.2 to 135.1-2019 - 7.3.2.21.3.2
Change BTL - 7.3.2.21.3.3 to 135.1-2019 - 7.3.2.21.3.3
Change BTL - 7.3.2.21.3.4 to 135.1-2019 - 7.3.2.21.3.4
Change BTL - 7.3.2.21.3.5 to 135.1-2019 - 7.3.2.21.3.5
Change BTL - 7.3.2.21.3.6 to 135.1-2019 - 7.3.2.21.3.6
Change BTL - 7.3.2.21.3.X7 to 135.1-2019 - 7.3.2.21.3.7 - Test name change 'test' vs 'Test'
Change BTL - 7.3.2.21.3.X8 to 135.1-2019 - 7.3.2.21.3.8
Change BTL - 7.3.2.21.3.X9 to 135.1-2019 - 7.3.2.21.3.9

Change 135.1-2013 - 7.3.2.24.18 - Records_Since_Notification to 135.1-2019 - 7.3.2.24.18 - Records_Since_Notification Test.

Change BTL-7.3.2.26 Remote Logging of Notifications to BTL-7.3.2.25.2 (same title).
Change BTL-7.3.2.27 Internal Logging of ACK_NOTIFICATIONs to BTL-7.3.2.25.3 (same title).
Change BTL-7.3.2.28 Remote Logging of ACK_NOTIFICATIONs to BTL-7.3.2.25.4 (same title).

Change 135.1-2013 - 8.4.3.1 - Numerical Algorithm (CHANGE_OF_VALUE Tests - ConfirmedEventNotification) to 135.1-2019 - 8.4.3.1 - Numerical Algorithm (ConfirmedEventNotification)
Change 135.1-2013 - 8.4.3.2 - Bitstring Algorithm (CHANGE_OF_VALUE Tests - ConfirmedEventNotification) to 135.1-2019 - 8.4.3.2 - Bitstring Algorithm (ConfirmedEventNotification)

Change BTL - 8.4.4 - COMMAND_FAILURE Tests to BTL - 8.4.4 - COMMAND_FAILURE Tests (ConfirmedEventNotification)

Change 135.1-2013 - 8.4.7 - BUFFER_READY Tests to 135.1-2019 - 8.4.7 -BUFFER_READY Tests (ConfirmedEventNotification)

Change BTL - 8.4.X1 to 135.1-2019 - 8.4.10
Change BTL - 8.4.X2 to 135.1-2019 - 8.4.11
Change BTL - 8.4.X3 to 135.1-2019 - 8.4.12
Change BTL - 8.4.X4 to 135.1-2019 - 8.4.13
Change BTL - 8.4.X7 to 135.1-2019 - 8.4.14 and change title from UNSIGNED_RANGE ConfirmedEventNotification Tests to UNSIGNED_RANGE Test (ConfirmedEventNotification Test).

Change BTL - 8.4.X8 to 135.1-2019 - 8.4.15

Change BTL - 8.4.X9.15 to BTL - 8.4.17.15

Change 135.1-2013 - 8.5.3.1 - Numerical Algorithm (CHANGE_OF_VALUE Tests - UnconfirmedEventNotification) to 135.1-2019 - 8.5.3.1 - Numerical Algorithm (UnconfirmedEventNotification)
Change 135.1-2013 - 8.5.3.2 - Bitstring Algorithm (CHANGE_OF_VALUE Tests - UnconfirmedEventNotification) to 135.1-2019 - 8.5.3.2 - Bitstring Algorithm (UnconfirmedEventNotification)
Change 135.1-2013 - 8.5.7 - BUFFER_READY Tests to 135.1-2019 - 8.5.7 -BUFFER_READY Tests (UnconfirmedEventNotification)

Change BTL - 8.5.X1 to 135.1-2019 - 8.5.10 Change Title to: DOUBLE_OUT_OF_RANGE Test (UnconfirmedEventNotification)
Change BTL - 8.5.X2 to 135.1-2019 - 8.5.11 Change Title to: SIGNED_OUT_OF_RANGE Test (UnconfirmedEventNotification)
Change BTL - 8.5.X3 to 135.1-2019 - 8.5.12 Change Title to: UNSIGNED_OUT_OF_RANGE Test (UnconfirmedEventNotification)
Change BTL - 8.5.X4 to 135.1-2019 - 8.5.13 Change Title to: CHANGE_OF_CHARACTERSTRING Test (UnconfirmedEventNotification)

Change BTL - 8.5.X6 to 135.1-2019 - 8.5.9 Change Title to: EXTENDED Test (UnconfirmedEventNotification)
Change BTL - 8.5.X8 to 135.1-2019 - 8.5.15.

Change

BTL - 8.5.X9.1 CHANGE_OF_RELIABILITY with No Fault Algorithm to
BTL - 8.5.17.1 CHANGE_OF_RELIABILITY with the NONE fault Algorithm (UnconfirmedEventNotification)

Change
BTL - 8.5.X9.2 CHANGE_OF_RELIABILITY with the FAULT_CHARACTERSTRING Algorithm
BTL - 8.5.17.2 CHANGE_OF_RELIABILITY with the FAULT_CHARACTERSTRING Algorithm (UnconfirmedEventNotifications)

Change
BTL - 8.5.X9.3 CHANGE_OF_RELIABILITY with the FAULT_EXTENDED Algorithm
BTL - 8.5.17.3 CHANGE_OF_RELIABILITY with the FAULT_EXTENDED Algorithm (UnconfirmedEventNotification)

Change
BTL - 8.5.X9.4 CHANGE_OF_RELIABILITY with the FAULT_LIFE_SAFETY Algorithm
BTL - 8.5.17.4 CHANGE_OF_RELIABILITY with the FAULT_LIFE_SAFETY Algorithm (UnconfirmedEventNotifications)

Change
BTL - 8.5.X9.5 CHANGE_OF_RELIABILITY with the FAULT_STATE Algorithm
BTL - 8.5.17.5 CHANGE_OF_RELIABILITY with the FAULT_STATE Algorithm (UnconfirmedEventNotifications)

Change
BTL - 8.5.X9.6 CHANGE_OF_RELIABILITY with the FAULT_STATUS_FLAGS Algorithm
BTL - 8.5.17.6 CHANGE_OF_RELIABILITY with the FAULT_STATUS_FLAGS Algorithm (UnconfirmedEventNotifications)

Change BTL - 8.5.X9.7.1 to BTL - 8.5.17.7.1
Change BTL - 8.5.X9.7.2 to BTL - 8.5.17.7.2
Change BTL - 8.5.X9.7.3 to BTL - 8.5.17.7.3

Change
BTL - 8.5.X9.8 CHANGE_OF_RELIABILITY of Event Enrollment Object, Monitored Object Fault
BTL - 8.5.17.8 CHANGE_OF_RELIABILITY of Event Enrollment Object, Monitored Object Fault (UnconfirmedEventNotifications)

Change
BTL - 8.5.X9.9 CHANGE_OF_RELIABILITY of Event Enrollment Object Fault
BTL - 8.5.17.9 CHANGE_OF_RELIABILITY of Event Enrollment Object Fault (UnconfirmedEventNotifications)

Change
BTL - 8.5.X9.10 After FAULT-to-NORMAL, Re-Notification of OFFNORMAL
BTL - 8.5.17.10 After FAULT-to-NORMAL, Re-Notification of OFFNORMAL (UnconfirmedEventNotifications)

[In BTL Test Plan, find all places where 8.5.17.* are used and add the corresponding reference 8.4.17.* to the required tests. Following the below example for all tests. See Sections: 5.2.30 – 35, 5.2.37, 5.2.42, 5.3.1, 5.3.18 - 23, 5.3.25, 5.3.30]

X.Y.Z Implements the CHANGE_OF_RELIABILITY - XXX

The IUT contains, or can be made to contain, an object that can generate EventNotifications with an Event_Type of CHANGE_OF_RELIABILITY and supports the specified algorithm.

BTL - 8.4.17.1 - CHANGE_OF_RELIABILITY with No Fault Algorithm (ConfirmedEventNotification)		
	Test Conditionality	The IUT contains, or can be made to contain, an object that can generate EventNotifications with an Event_Type of CHANGE_OF_RELIABILITY and which does not apply a standardized fault algorithm.

	Test Directives	Apply this test to all object types that support fault detection but do not apply a standardized fault algorithm.
	Testing Hints	
BTL - 8.5.17.1 - CHANGE_OF_RELIABILITY with No Fault Algorithm (UnconfirmedEventNotification)		
	Test Conditionality	The IUT contains, or can be made to contain, an object that can generate EventNotifications with an Event_Type of CHANGE_OF_RELIABILITY and which does not apply a standardized fault algorithm.
	Test Directives	Apply this test to all object types that support fault detection but do not apply a standardized fault algorithm.
	Testing Hints	

Change BTL - 9.1.1.2 to 135.1-2019 - 9.1.1.2
Change BTL - 9.1.1.3 to 135.1-2019 - 9.1.1.3
Change BTL - 9.1.1.4 to 135.1-2019 - 9.1.1.4
Change BTL - 9.1.1.5 to 135.1-2019 - 9.1.1.5
Change BTL - 9.1.1.6 to 135.1-2019 - 9.1.1.6
Change BTL - 9.1.1.8 to 135.1-2019 - 9.1.1.8
Change BTL - 9.1.1.9 to 135.1-2019 - 9.1.1.9
Change BTL - 9.1.1.10 to 135.1-2019 - 9.1.1.10
Change BTL - 9.1.1.11 to 135.1-2019 - 9.1.1.11
Change BTL - 9.1.1.X3 to 135.1-2019 - 9.1.1.14 and change title by appending "**, Revision 5 or higher only**"
Change BTL - 9.1.X1 to 135.1-2019 - 9.1.1.15 and change title to " **Unsupported Message Text Character Set AcknowledgeAlarm Test** "

Change BTL - 9.1.2.1 to 135.1-2019 - 9.1.2.1

Change BTL - 9.1.2.3 to 135.1-2019 - 9.1.2.3
Change BTL - 9.1.2.4 to 135.1-2019 - 9.1.2.4
Change BTL - 9.1.2.5 to 135.1-2019 - 9.1.2.5
Change BTL - 9.1.2.6 to 135.1-2019 - 9.1.2.6 and change title to " **Unsuccessful Alarm Acknowledgment of Unconfirmed Event Notifications Because the 'Event ObjectIdentifier' is Invalid**"

Change BTL - 9.1.2.7 to 135.1-2019 - 9.1.2.7

Change 135.1-2013 - 9.4.1 - ConfirmedEventNotification Using the Time Form of the 'Timestamp' Parameter and Conveying a Text Message to 135.1-2019 - 9.4.1 - ConfirmedEventNotification Using the Time Form of the 'Timestamp' Parameter and Conveying a Message Text

Change 135.1-2013 - 9.4.2 - ConfirmedEventNotification Using the DateTime Form of the 'Timestamp' Parameter and no Text Message to 135.1-2019 - 9.4.2 - ConfirmedEventNotification Using the DateTime Form of the 'Timestamp' Parameter and no Message Text

Change 135.1-2013 - 9.4.3 - ConfirmedEventNotification Using the Sequence Number Form of the 'Timestamp' Parameter and no Text Message to 135.1-2019 - 9.4.3 - ConfirmedEventNotification Using the Sequence Number Form of the 'Timestamp' Parameter and no Message Text

Change BTL - 9.4.5 to 135.1-2019 - 9.4.5
Change BTL - 9.4.6 to 135.1-2019 - 9.4.6
Change BTL - 9.4.X1 to 135.1-2019 - 9.4.7
Change BTL - 9.5.X1 to 135.1-2019 - 9.5.3
Change BTL - 9.7.1.1 to 135.1-2019 - 9.7.1.1
Change BTL - 9.7.2.3 to 135.1-2019 - 9.7.2.3
Change BTL - 9.8.6 to 135.1-2019 - 9.8.6
Change BTL - 9.11.2.2 to 135.1-2019 - 9.11.2.2
Change BTL - 9.16.2.1 to 135.1-2019 - 9.16.2.1
Change BTL - 9.16.2.6 to 135.1-2019 - 9.16.2.6
Change BTL - 9.20.1.7 to 135.1-2019 - 9.20.1.7

Change BTL - 9.20.1.9 to 135.1-2019 - 9.20.1.9
Change BTL - 9.31.1.2 to 135.1-2019 - 9.31.1.2
Change BTL - 9.33.1.3 to 135.1-2019 - 9.33.1.3

Change BTL - 13.X13.1 to BTL - 13.9.X1

BTL Specified Tests
Note: We are deleting approx. 63 tests from BTL Specified Tests (out of a total of 503 tests)

Move all Notes to Tester to above Test Steps.

[In BTL Specified Tests, update Section 5 EPICS to contain 135-2019 version + new (p) added in 16.1fix1]

[In BTL Specified Test, update test 7.1.1 to derive from 135.1-2019 as shown below.]

7.1.1 Read Support Test Procedure
Reason for Change: Updated the error codes allowed if prior to Protocol_Revision 13. Added Explanatory Notes To Tester for using with ReadPropertyMultiple. Moved line from Purpose to Test Concept. Added 'server' entry in Abort response.

~~Dependencies:ReadProperty Service Execution Tests, 9.18.~~

Purpose: Verifies that a correct response is returned when each property of each object is read using BACnet ReadProperty and ReadPropertyMultiple services. ~~The test is performed once using ReadProperty and once using ReadPropertyMultiple, if supported. When verifying array properties, whole arrays shall be read without using an ARRAY INDEX, where possible.~~

Test Concept: The test is performed once using ReadProperty and once using ReadPropertyMultiple, if supported. When verifying array properties, the whole array shall be read without using an array index, where possible.

Test Steps:

```
1.  REPEAT X = (all objects in the IUT's database) DO {
        REPEAT Y = (all properties in object X) DO {
            IF (Y is defined by the standard as not accessible by ReadProperty Services) THEN
                TRANSMIT ReadProperty-Request,
                    'Object Identifier' = X,
                    'Property Identifier' = Y
            IF (Protocol_Revision >= 13~~7~~) THEN
                RECEIVE BACnet-Error PDU,
                    'Error Class' = PROPERTY,
                    'Error Code' = READ_ACCESS_DENIED
            ELSE
                RECEIVE BACnet-Error PDU,
                    'Error Class' = OBJECT | PROPERTY,
                    'Error Code' = (any of the error codes for an OBJECT or PROPERTY class)
                    ~~Error Class = PROPERTY,~~
                    ~~Error Code = READ_ACCESS_DENIED | OTHER~~
                    ~~| (BACnet-Error-PDU,~~
                    ~~Error Class - OBJECT,~~
                    ~~Error Code = OTHER),~~
                    | (BACnet-Error-PDU,
                    ~~Error Class~~ 'Error Class' = SERVICES,
                    ~~Error Code~~ 'Error Code' = SERVICE_REQUEST_DENIED | OTHER)

        ELSE IF (Y is an array and its value is too long to return given the IUT's APDU and segmentation limitations)
THEN
            TRANSMIT ReadProperty-Request,
                'Object Identifier' = X,
                'Property Identifier' = Y
            RECEIVE BACnet-Abort-PDU,
                'Server' = TRUE,
                'Abort Reason' = SEGMENTATION_NOT_SUPPORTED
                                | BUFFER_OVERFLOW
            TRANSMIT ReadProperty-Request,
                'Object Identifier' = X,
                'Property Identifier' = Y,
                'Property Array Index' = 0
        RECEIVE ReadProperty-ACK,
            'Object Identifier' = X,
            'PropertyIdentifier' = Y,
            'Property Array Index' = 0,
            'Property Value' = (N: the number of array elements in Y as indicated in the EPICS)
        REPEAT Z = (1 .. N) DO {
            VERIFY (X), Y = (the value for element Z as indicated in the EPICS), ARRAY INDEX = Z
        }
    ELSE
        VERIFY (X), Y = (the value for this property specified in the EPICS)
        }
    }
```

Notes to Tester: For cases where the EPICS indicates that the value of a property is unspecified using the "?" symbol, any value that is of the correct datatype shall be considered to be a match. *When using the ReadPropertyMultiple service, a received ReadPropertyMultiple-ACK containing the specified Error Class and Error Code shall also be considered a Passing result.*

This test was modified again in Addenda Fix1
~~Delete test 7.1.X3.~~

Section 7.3.1.7 COV Tests, add a strike through of "Tests to demonstrate COV functionality are covered in 8.2 and 9.6" which is what exists in 135-1-2019.

[In BTL Specified Tests, update the title of this test and change tracking to match below derived from 135.1-2019.]

7.3.1.10.1 Event_Enable Test for TO_OFFNORMAL and TO_NORMAL, and TO_FAULT

Reason for Change: This test was modified to add clarifying sentence to Purpose and clarifying sentence to "Notes to Tester".

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 objects that support intrinsic reporting.*

Test Concept: The IUT is configured with an event-generating object, O1, such that the Event_Enable property is tested in all supported states. 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: If the Event_Enable property is configurable, repeat the test with Event_Enable=(T,F,F),(F,T,F),(F,F,T). If the Event_Enable property is not configurable, then follow the test steps as written and verify correct behavior for the value of the Event_Enable property. All other properties in O1, and any supporting objects, shall be configured to allow these events to be generated. The event-generating object shall be in a NORMAL state at the start of the test. D1 is either the pTimeDelay parameter or, in the case of the EXTENDED and proprietary algorithms, a vendor specific delay (may be zero).

D2 is either the pTimeDelayNormal parameter or, in case of the EXTENDED and proprietary algorithms, a vendor specific delay (may be zero).

```
1.  VERIFY pCurrentState~~-~~= NORMAL
2.  WAIT (pTimeDelay + **Notification Fail Time**)
3.  IF (O1 contains pFeedbackValue) THEN
        MAKE (pFeedbackValue differ from pMonitoredValue)
    ELSE IF (pMonitoredValue is writable) THEN
        WRITE pMonitoredValue = (a value that is OFFNORMAL)
    ELSE
        MAKE (pMonitoredValue have a value that is OFFNORMAL)
4.  WAIT (D1)
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' =         O1,
                'Time Stamp' =                      (any valid time stamp),
                '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),
                'Message Text' =                    (optional, any valid message text),
                'Notify Type' =                     EVENT | ALARM,
                'AckRequired' =                     TRUE | FALSE,
                'From State' =                      NORMAL,
                'To State' =                        OFFNORMAL,
                'Event Values' =                    (values appropriate to the event type)
            TRANSMIT BACnet-SimpleACK-PDU
        }
        ELSE
            CHECK (verify that the IUT did not transmit an event notification message)
6.  VERIFY pCurrentState = OFFNORMAL
7.  IF (O1 contains pFeedbackValue) THEN
        MAKE (pFeedbackValue equal to pMonitoredValue)
    ELSE IF (pMonitoredValue is writable) THEN
        WRITE pMonitoredValue = (a value that is NORMAL)
    ELSE
        MAKE (pMonitoredValue have a value that is NORMAL)
8.  WAIT (D2)
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' =         O1,
                'Time Stamp' =                      (any valid time stamp),
                '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),
                'Message Text' =                    (optional, any valid message text),
                'Notify Type' =                     EVENT | ALARM,
                'AckRequired' =                     TRUE | FALSE,
                'From State' =                      OFFNORMAL,
                'To State' =                        NORMAL,
                'Event Values' =                    (values appropriate to the event type)
            TRANSMIT BACnet-SimpleACK-PDU
        }
        ELSE
            CHECK (verify that the IUT did not transmit an event notification message)
10. VERIFY pCurrentState = NORMAL
11. IF (O1-can be placed into a fault condition) THEN {
        MAKE (a condition exist that will cause O1 to generate a TO-FAULT transition)
        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' =         *O1*~~(the event-generating object configured for this test),~~
                    'Time Stamp' =                      (any valid time stamp),
                    '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),
                    'Message Text' =                    (optional, any valid message text),
                    'Notify Type' =                 EVENT | ALARM,
                    'AckRequired' =                 TRUE | FALSE,
                    'From State' =                  NORMAL,
                    'To State' =                    FAULT,
                    'Event Values' =                (values appropriate to the event type)
                TRANSMIT BACnet-SimpleACK-PDU
            }
            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. in which case the TD shall skip all of the steps in which a BACnet-SimpleACK-PDU is sent.~~

Notes to Tester: The UnconfirmedEventNotification service may be substituted for the ConfirmedEventNotification service, in which case, the TD shall skip sending the BACnetSimpleACK-PDU messages after the receipt of the notifications.

[In BTL Specified Tests, delete the following tests]
Delete 7.3.1.10.2

[In BTL Specified Tests, replace the existing 7.3.1.11 test with the one shown here that is derived from 135.1-2019.]

7.3.1.11 Acked_Transitions Tests

Reason for Change: Corrected errata issues that are in 135.1-2019. Improved the text for Notes To Tester.

Purpose: To verify that the Acked_Transitions property tracks whether or not an acknowledgment has been received for a previously issued event notification. It also verifies the interrelationship between Status_Flags and Event_State.

Test Concept: The IUT is configured such that the Event_Enable property indicates that all event transitions are to trigger an event notification. The Acked_Transitions property shall have the value (TRUE, TRUE, TRUE) indicating that all previous transitions have been acknowledged. Each event transition is triggered and the Acked_Transitions property is monitored to verify that the appropriate bit is cleared when a notification message is transmitted and reset if an acknowledgment is received.

Configuration Requirements: The Event_Enable and Acked_Transitions properties shall be configured with a value of (TRUE, TRUE, TRUE). 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. The value of the Transitions parameter for all recipients shall be (TRUE, TRUE, TRUE).

Test Steps:

```
1.  VERIFY pCurrentState = NORMAL
2.  VERIFY Acked_Transitions = (TRUE, TRUE, TRUE)
3.  IF (Protocol_Revision is present AND Protocol_Revision >= 13) THEN
        VERIFY pStatusFlags = (FALSE, FALSE, ?, ?)
4.  IF (pMonitoredValue is writable) THEN
        WRITE pMonitoredValue = (a value that is OFFNORMAL)
    ELSE
        MAKE (pMonitoredValue have a value that is OFFNORMAL)
5.  WAIT (pTimeDelay)
6.  BEFORE **Notification Fail Time**
        RECEIVE ConfirmedEventNotification-Request,
            'Process Identifier' =              (PI1: any valid process ID),
            'Initiating Device Identifier' =    IUT,
            'Event Object Identifier' =         (the event-generating object configured for this test),
            'Time Stamp' =                      (Toffnormal: any valid time stamp),
            'Notification Class' =            (the class corresponding to the object being tested),
            'Priority' =                      (Poffnormal: the value configured to correspond to a TO-OFFNORMAL
transition),
            'Event Type' =                    (any valid event type),
            'Message Text' =                  (optional, any valid message text),
            'Notify Type' =                   (the notify type configured for this event),
            'AckRequired' =                   TRUE,
            'From State' =                    NORMAL,
            'To State' =                      OFFNORMAL,
            'Event Values' =                  (values appropriate to the event type)
7.  TRANSMIT BACnet-SimpleACK-PDU
8.  VERIFY pCurrentState = OFFNORMAL
9.  VERIFY Acked_Transitions = (FALSE, TRUE, TRUE)
10. IF (Protocol_revision is present AND Protocol_Revision >= 13 THEN
        VERIFY pStatusFlags = (TRUE, FALSE, ?, ?)
11. IF (pMonitoredValue is writable) THEN
        WRITE pMonitoredValue = (a value that is NORMAL)
    ELSE
        MAKE (pMonitoredValue have a value that is NORMAL)
12. WAIT (pTimeDelayNormal)
13. BEFORE **Notification Fail Time**
        RECEIVE ConfirmedEventNotification-Request,
            'Process Identifier' =            (PI2: any valid process ID),
            'Initiating Device Identifier' =  IUT,
            'Event Object Identifier' =       (the event-generating object configured for this test),
            'Time Stamp' =                    (Tnormal: any valid time stamp),
            'Notification Class' =            (the class corresponding to the object being tested),
            'Priority' =                      (Pnormal: the value configured to correspond to a TO-NORMAL transition),
            'Event Type' =                    (any valid event type),
            'Message Text' =                  (optional, any valid message text),
            'Notify Type' =                   (the notify type configured for this event),
            'AckRequired' =                   TRUE,
            'From State' =                    OFNORMAL,
            'To State' =                      NORMAL,
            'Event Values' =                  (values appropriate to the event type)
14. TRANSMIT BACnet-SimpleACK-PDU
15. VERIFY pCurrentState = NORMAL
16. VERIFY Acked_Transitions = (FALSE, TRUE, FALSE)
17. IF (Protocol_Revision is present AND Protocol_Revision >= 13) THEN
        VERIFY pStatusFlags = (FALSE, FALSE, ?,?)
18. IF (the event-triggering object can be placed into a fault condition) THEN {
        MAKE (a condition exist that will cause the object to generate a fault condition)
        BEFORE **Notification Fail Time**
            RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' =            (PI3: any valid process ID),
                'Initiating Device Identifier' =  IUT,
                'Event Object Identifier' =       (the event-generating object configured for this test),
                'Time Stamp' =                    (Tfault: any valid time stamp),
                'Notification Class' =            (the class corresponding to the object being tested),
                'Priority' =                      (Pfault: the value configured to correspond to a TO-FAULT transition),
                'Event Type' =                    IF (Protocol_Revision < 13) THEN
                                                      (any valid event type),
                                                  ELSE
                                                      CHANGE_OF_RELIABILITY,
                'Message Text' =                  (optional, any valid message text),
                'Notify Type' =                   (the notify type configured for this event),
                'AckRequired' =                   TRUE,
                'From State' =                    NORMAL,
                'To State' =                      FAULT,
                'Event Values' =                  (values appropriate to the event type)
        TRANSMIT BACnet-SimpleACK-PDU
        VERIFY pCurrentState = FAULT
        VERIFY Acked_Transitions = (FALSE, FALSE, FALSE)
        TRANSMIT AcknowledgeAlarm-Request,
            'Acknowledging Process Identifier' =    (PI3),
            'Event Object Identifier' =             (the event-generating object configured for this test),
            'Event State Acknowledged' =            FAULT,
            'Acknowledgement Source' =              (a character string),
            'Time Stamp' =                          (Tfault),
            'Time of Acknowledgment' =              (the TD's current time)
        RECEIVE BACnet-SimpleACK-PDU
        IF (Protocol_Revision is present AND Protocol_Revision ≥ 1) THEN
            BEFORE **Notification Fail Time**
                RECEIVE ConfirmedEventNotification-Request,
                    'Process Identifier' =          (PI3),
                    'Initiating Device Identifier' = IUT,
                    'Event Object Identifier' =     (the event-generating object configured for this test),
                    'Time Stamp' =                  (Tfault *the current time or sequence number*),
                    'Notification Class' =          (the class corresponding to the object being tested),
                    'Priority' =                    (Pfault),
                    'Event Type' =                  IF (Protocol_Revision < 13)
                                                            (any valid event type),
                                                    ELSE
                                                            CHANGE_OF_RELIABILITY,
                    'Message Text' =                (optional, any valid message text),
                    'Notify Type' =                 ACK_NOTIFICATION,
                    'To State' =                    FAULT
        ELSE
            BEFORE **Notification Fail Time**
                RECEIVE ConfirmedEventNotification-Request,
                    'Process Identifier' =          (PI3),
                    'Initiating Device Identifier' = IUT,
                    'Event Object Identifier' =     (the event-generating object configured for this test),
                    'Time Stamp' =                  (Tfault *the current time or sequence number*),
                    'Notification Class' =          (the class corresponding to the object being tested),
                    'Priority' =                    (Pfault),
                    'Event Type' =                  (any valid event type),
                    'Notify Type' =                 ACK_NOTIFICATION
        TRANSMIT BACnet-SimpleACK-PDU
        VERIFY Acked_Transitions = (FALSE, TRUE, FALSE)
    }
19. TRANSMIT AcknowledgeAlarm-Request,
        'Acknowledging Process Identifier' =        (PI2),
        'Event Object Identifier' =                 (the event-generating object configured for this test),
        'Event State Acknowledged' =                NORMAL,
        'Time Stamp' =                              (Tnormal),
        'Acknowledgement Source' =                  (a character string),
        'Time of Acknowledgment' =                  (the TD's current time)
20. RECEIVE BACnet-SimpleACK-PDU
21. IF (Protocol_Revision is present and Protocol_Revision ≥ 1) THEN
        BEFORE **Notification Fail Time**
            RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' =              (PI2),
                'Initiating Device Identifier' =    IUT,
                'Event Object Identifier' =         (the event-generating object configured for this test),
                'Time Stamp' =                      (Tnormal),
                'Notification Class' =              (the class corresponding to the object being tested),
                'Priority' =                        (Pnormal),
                'Event Type' =                      (any valid event type),
                'Notify Type' =                     ACK_NOTIFICATION,
                'To State' =                       NORMAL
    ELSE
        BEFORE **Notification Fail Time**
            RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' =             (PI2),
                'Initiating Device Identifier' =   IUT,
                'Event Object Identifier' =        (the event-generating object configured for this test),
                'Time Stamp' =                     (Tnormal),
                'Notification Class' =             (the class corresponding to the object bind tested),
                'Priority' =                       (Pnormal),
                'Event Type' =                     (any valid event type),
                'Message Text' =                   (optional, any valid message text),
                'Notify Type' =                    ACK_NOTIFICATION
22. TRANSMIT BACnet-SimpleACK-PDU
23. VERIFY Acked_Transitions = (FALSE, TRUE, TRUE)
24. TRANSMIT AcknowledgeAlarm-Request,
        'Acknowledging Process Identifier' =   (PI1),
        'Event Object Identifier' =            (the event-generating object configured for this test),
        'Event State Acknowledged' =           OFFNORMAL,
        'Time Stamp' =                         (Toffnormal),
        'Acknowledgement Source' =             (a character string),
        'Time of Acknowledgment' =             (the TD's current time)
25. RECEIVE BACnet-SimpleACK-PDU
26. IF (Protocol_Revision is present and Protocol_Revision ≥ 1) THEN
        BEFORE **Notification Fail Time**
            RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' =             (PI1),
                'Initiating Device Identifier' =   IUT,
                'Event Object Identifier' =        (the event-generating object configured for this test),
                'Time Stamp' =                     (Toffnormal),
                'Notification Class' =             (the class corresponding to the object being tested),
                'Priority' =                       (Poffnormal),
                'Event Type' =                     (any valid event type),
                'Message Text' =                   (optional, any valid message text),
                'Notify Type' =                    ACK_NOTIFICATION,
                'To State' =                       OFFNORMAL
    ELSE
        BEFORE **Notification Fail Time**
            RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' =             (PI1),
                'Initiating Device Identifier' =   IUT,
                'Event Object Identifier' =        (the event-generating object configured for this test),
                'Time Stamp' =                     (Toffnormal *the current time or sequence number*),
                'Notification Class' =             (the class corresponding to the object being tested),
                'Priority' =                       (Poffnormal),
                'Event Type' =                     (any valid event type),
                'Message Text' =                   (optional, any valid message text),
                'Notify Type' =                    ACK_NOTIFICATION
27. TRANSMIT BACnet-SimpleACK-PDU
28. VERIFY Acked_Transitions = (TRUE, TRUE, TRUE)
```

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 BACnet-SimpleACK-PDU is sent~~ *sending the BACnet-SimpleACK-PDU messages after receiving the notifications*.

[In BTL Specified Tests, delete the following tests.]
Delete 7.3.1.X4
Delete 7.3.1.X5
Delete 7.3.1.X6.1
Delete 7.3.1.X6.2
Delete 7.3.1.X6.3

Delete 7.3.1.X7.1
Delete 7.3.1.X7.2
Delete 7.3.1.X8.1
Delete 7.3.1.X8.2
Delete 7.3.1.X9.1
Delete 7.3.1.X9.2
Delete 7.3.2.9.7
Delete 7.3.2.21.3.1
Delete 7.3.2.21.3.2
Delete 7.3.2.21.3.3
Delete 7.3.2.21.3.4
Delete 7.3.2.21.3.5
Delete 7.3.2.21.3.6
Delete 7.3.2.21.3.X7
Delete 7.3.2.21.3.X8
Delete 7.3.2.21.3.X9

[In BTL Specified Tests, Move 7.3.2.23.3.7 to the correct location in the document.]

[In BTL Specified Tests, replace current version with the version shown below.]

7.3.2.25.1 Internal Logging of Notifications

Reason for Change: Fixed 'Result Flags' in step 12.

Purpose: To verify the IUT correctly collects and represents the Notifications which it initiates.

Test Concept: Make the IUT generate two event notification messages which the IUT logs. Use ReadRange to retrieve them from an Event Log and compare the two representations.

Configuration Requirements: The tester shall choose two events which are configured to be sent to the TD and to be placed into one of the IUT's Event Logs, LO1.

Test Steps:

1. WRITE Enable = TRUE
2. MAKE (a condition exist that will cause the device to generate an event transition)
3. WAIT D1
4. RECEIVE ConfirmedEventNotification-Request,

'Process Identifier' =	(any valid process identifier),
'Initiating Device Identifier' =	IUT,
'Event Object Identifier' =	(any valid object),
'Time Stamp' =	(T1, any valid timestamp),
'Notification Class' =	(any valid notification class),
'Priority' =	(any valid priority),
'Event Type' =	(any standard event type),
'Message Text' =	(optional, any valid message text),
'Notify Type' =	ALARM | EVENT,
'AckRequired' =	TRUE | FALSE,
'From State' =	(state S1, any valid state for this event type),
'To State' =	(state S2, any valid state for this event type that can follow S1),
'Event Values' =	(any values appropriate to the event type)

5. TRANSMIT BACnet-SimpleACK-PDU
6. MAKE (a condition exist that will cause the device to generate an event transition)
7. WAIT D2
8. RECEIVE ConfirmedEventNotification-Request,

'Process Identifier' =	(any valid process identifier),
'Initiating Device Identifier' =	IUT,
'Event Object Identifier'	(any valid object),
'Time Stamp' =	(T2, any valid timestamp),

```
        'Notification Class' =          (any valid notification class),
        'Priority' =                    (any valid priority),
        'Event Type' =                  (any standard event type),
        'Message Text' =                (optional, any valid message text),
        'Notify Type' =                 ALARM | EVENT,
        'AckRequired' =                 TRUE | FALSE,
        'From State' =                  (state S3, any valid state for this event type),
        'To State' =                    (state S4, any valid state for this event type that can follow S3),
        'Event Values' =                (any values appropriate to the event type)
9.  TRANSMIT BACnet-SimpleACK-PDU
10. READ RC = LO1, Record_Count
11. TRANSMIT ReadRange-Request,
        'Object Identifier' =           LO1,
        'Property Identifier' =         Log_Buffer,
        'Reference Index'=              RC,
        'Count' =                       -2
12. RECEIVE ReadRange-ACK,
            'Object Identifier' =       LO1,
            'Property Identifier' =     Log_Buffer,
            'Result Flags' =            {FALSE, ?, FALSE~~TRUE~~},
            'Item Count'    =           2,
            'Item Data' =               (logged data that matches the information received in steps 3 and 6,
                                        except that Process_Identifier may be any value and is not required to match)
11. CHECK (T2 > T1, and that the notifications were logged in order)
```

Notes to Tester: When the UnconfirmedEventNotification service is used instead of the ConfirmedEventNotification service, the TD shall skip the steps in which a SimpleACK-PDU is sent.

[In BTL Specified Tests, Change 7.3.2.26 to 7.3.2.25.2 and Add a Reason for Change]
[In BTL Specified Tests, Change 7.3.2.27 - Internal Logging of ACK_NOTIFICATIONs to 7.3.2.25.3, and add a reason for Change]
[In BTL Specified Tests, Change 7.3.2.28 - Remote Logging of ACK_NOTIFICATIONs to 7.3.2.25.4, and add a reason for Change.]

[In BTL Specified Tests, Change 7.3.2.30 section title from Alert Enrollment to Notification Forwarder Object Tests]

[In BTL Specified Tests, Test 7.3.2.30.6 needs a reason for change.]

[In BTL Specified Tests, Change 7.3.2.X54.22 - Lighting Output Present Value between 0.0 and 1.0 Test and Change 7.3.2.X54.21 - Lighting Output Tracking Test title to remove the '-']
]

[In BTL Specified Tests, Test 8.1 needs a reason for Change.]

[In BTL Specified Tests, Update test 8.2.1 to new change marking revised for 135.1-2019 as shown below. Note these changes include those found in Addenda Fix1]

8.2.1 Change of Value Notification from an Analog Input, Analog Output, and Analog Value Object for Changes to Present_Value Property in Objects with a COV_Increment

Reason for Change: Updated description of the 'List of Values' to improve readability. Updated 'Configuration Requirements'. Add clarification to test that the last COVNotification shall reflect the correct values. Removed unnecessary RECEIVE BACnet-SimpleACK-PDU steps. Improved test name and wording to include generic object references.

Purpose: To verify that the IUT can initiate ConfirmedCOVNotification service requests conveying a change of the Present_Value property ~~of~~ ~~Analog Input, Analog Output, and Analog Value~~ *in* objects *that support COV_Increment.*

Test Concept: A subscription for COV notifications is established, using a Lifetime of L. L shall be set to a value less than 24 hours and large enough to complete the test. The Present_Value of the monitored object is changed by an amount less than the COV increment and it is verified that no COV notification is received. The Present_Value is then changed by an amount greater than the COV increment and a notification shall be received. The Present_Value may be changed using the WriteProperty service or by another means such as changing the input signal represented by an Analog Input object. For some implementations it may be necessary to write to the Out_Of_Service property first to accomplish this task. For implementations where it is not possible to write to these properties at all the vendor shall provide an alternative trigger mechanism to accomplish this task. All of these methods are equally acceptable.

Configuration Requirements: At the beginning of the test, the Out_Of_Service property shall have a value of FALSE. *Select an object where Present_Value is not expected to change outside the tester's control by more than COV_Increment or which has a writable Out_Of_Service.*

Notes to Tester: The IUT may initiate additional COVNotifications. The final COVNotification shall accurately reflect Present_Value and Status_Flags.

Test Steps:

```
REPEAT X = (one supported object of each type ~~from the set Analog Input, Analog Output, and Analog Value~~) DO {
1.  TRANSMIT SubscribeCOV-Request,
        'Subscriber Process Identifier' =       (any value > 0 chosen by the TD),
        'Monitored Object Identifier' =         X,
        'Issue Confirmed Notifications' =       TRUE,
        'Lifetime' =                            L
2.  RECEIVE BACnet-SimpleACK-PDU
3.  BEFORE **Notification Fail Time**
      RECEIVE ConfirmedCOVNotification-Request,
        'Subscriber Process Identifier' =       (the same value used in step 1),
        'Initiating Device Identifier' =        IUT,
        'Monitored Object Identifier' =         X,
        'Time Remaining' =                      (any value appropriate for the Lifetime selected),
        'List of Values' =                      (the initial Present_Value and initial Status_Flags)
4.  TRANSMIT BACnet-SimpleACK-PDU
5.  TRANSMIT ReadProperty-Request,
        'Object Identifier' =                   X,
        'Property Identifier' =                 COV_Increment
6.  RECEIVE BACnet-ComplexACK-PDU,
        'Object Identifier' =                   X,
        'Property Identifier' =                 COV_Increment,
        'Property Value' =                      (a value "increment" that will be used below)
7.  IF (Out_Of_Service is writable) THEN
        WRITE X, Out_Of_Service = TRUE
        ~~RECEIVE BACnet-SimpleACK-PDU~~
        BEFORE **Notification Fail Time**
            RECEIVE ConfirmedCOVNotification-Request,
                'Subscriber Process Identifier' = (the same value used in step 1),
                'Initiating Device Identifier' =  IUT,
                'Monitored Object Identifier' =   X,
                'Time Remaining' =                (any value appropriate for the Lifetime selected),
                'List of Values' =                (*ReportedPV* ~~=any value appropriate for~~ *the current* Present_Value, ~~and new~~ Status_Flags)
        TRANSMIT BACnet-SimpleACK-PDU
8.  IF (Present_Value is now writable) THEN
        WRITE X, Present_Value = (any value that differs ~~from "initial Present_Value"~~ *ReportedPV* by less than "increment")
        ~~RECEIVE BACnet-SimpleACK-PDU~~
    ELSE
        MAKE (Present_Value  = any value that differs from ~~"initial Present_Value"~~ *ReportedPV* by less than "increment")
9.  WAIT **Notification Fail Time**
10. CHECK (verify that no COV notification was transmitted)
```

```
11. IF (Present_Value is now writable) THEN
        WRITE X, Present_Value = (any value that differs from ~~"initial Present_Value"~~ *ReportedPV* by an amount greater than "increment")
        ~~RECEIVE BACnet-SimpleACK-PDU~~
    ELSE
        MAKE (Present_Value  = any value that differs from ~~"initial Present_Value"~~ *ReportedPV* by an amount greater than "increment")
12. BEFORE **NotificationFailTime**
        RECEIVE ConfirmedCOVNotification-Request,
            'Subscriber Process Identifier' =     (the same value used in step 1),
            'Initiating Device Identifier' =      IUT,
            'Monitored Object Identifier' =       X,
            'Time Remaining' =                    (any value appropriate for the Lifetime selected),
            'List of Values' =                    (the new Present_Value and new Status_Flags)
13. TRANSMIT BACnet-SimpleACK-PDU
14. TRANSMIT SubscribeCOV-Request,
        'Subscriber Process Identifier' =         (the same value used in step 1),
        'Monitored Object Identifier' =           X
15. RECEIVE BACnet-SimpleACK-PDU
16. IF (Out_Of_Service is writable) THEN
        WRITE X, Out_Of_Service =                 FALSE
        ~~RECEIVE BACnet-SimpleACK-PDU~~
```

[In BTL Specified Tests, Test 8.4.4 needs reason for change.]

[In BTL Specified Tests, Change Title of 8.4.4 - COMMAND_FAILURE Tests to 8.4.4 - COMMAND_FAILURE Tests (ConfirmedEventNotification)]

[In BTL Specified Tests, delete the following tests. The only change to these tests was the Purpose statement and BTL-WG voted to drop our changes and keep the SSPC version.]

Delete 8.4.X1
Delete 8.4.X2
Delete 8.4.X3
Delete 8.4.X4
Delete 8.4.X7
Delete 8.4.X8

[In BTL Specified Tests, Change 8.4.X6 to 8.4.9. Change title, replace test with content here which derives from 135.1-2019.]

8.4.9 EXTENDED Tests (ConfirmedEventNotifications)

Reason for Change: Added clarifying statement in Test Concept. Added Notes to Tester.

Purpose: To verify the correct generation EXTENDED event notifications.

Test Concept: The event generating object is made to transition to any state by any means necessary. The resulting ConfirmedEventNotification message is received and verified. *The object begins the test in a NORMAL state.*

Configuration Requirements: The IUT shall be configured such that the Event_Enable property has a value of TRUE for whichever transition shall be used for the test. The 'Issue_Confirmed_Notifications' parameter shall have a value of TRUE. D1 and D2 are vendor specified delays (either of them or both may be zero).

Test Steps:

```
1.  IF (the object generates TO-OFFNORMAL transitions) THEN {
        READ CS1 = pCurrentState
        MAKE (an OFFNORMAL condition exist)
        WAIT (D1)
        BEFORE **Notification Fail Time**
            RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' =              (any valid process ID),
                'Initiating Device Identifier' =    IUT,
                'Event Object Identifier' =         (the event generating object),
                'Time Stamp' =                      (TS1: the current local time),
                'Notification Class' =              (the configured notification class),
                'Priority' =                        (the value configured for TO_OFFNORMAL),
                'Event Type' =                      EXTENDED,
                'Message Text' =                    (optional, any valid message text),
                'Notify Type' =                     EVENT | ALARM,
                'AckRequired' =                     TRUE | FALSE,
                'From State' =                      CS1,
                'To State' =                        (CS2: any offnormal valid event state),
                'Event Values' =                    (pVendorId: any valid vendor id),
                                                    (pEventType: any valid event-type),
                                                    (a list of 0 or more valid parameters as defined by the Vendor)
        TRANSMIT BACnet-SimpleACK-PDU
        IF (Protocol_Revision is present AND Protocol_Revision >= 13) THEN
                VERIFY pStatusFlags = (TRUE, FALSE, ?, ?)
        VERIFY pCurrentState = CS2
        IF (ProtocolRevision is present and Protocol_Revision >= 1) THEN
                VERIFY Event_Time_Stamps = (TS1, *, *)
}

IF (the object generates TO_NORMAL transitions) THEN
    READ CS1 = Event_State
    MAKE (a NORMAL condition exist)
    WAIT D2
    BEFORE **Notification Fail Time**
        RECEIVE ConfirmedEventNotification-Request,
            'Process Identifier' =              (any valid process ID),
            'Initiating Device Identifier' =    IUT,
            'Event Object Identifier' =         (the intrinsic reporting object being tested),
            'Time Stamp' =                      (TS2: the current local time),
            'Notification Class' =              (the configured notification class),
            'Priority' =                        (the value configured for TO-NORMAL),
            'Event Type' =                      EXTENDED,
            'Message Text' =                    (optional, any valid message text),
            'Notify Type' =                     EVENT | ALARM,
            'AckRequired' =                     TRUE | FALSE,
            'From State' =                      CS1,
            'To State' =                        NORMAL,
            'Event Values' =                    (pVendorId: any valid vendor id),
                                                (pEventType: any valid event-type),
                                                (a list of 0 or more valid parameters as defined by the Vendor)

    TRANSMIT BACnet-SimpleACK-PDU
    IF (Protocol_Revision is present AND Protocol_Revision >= 13) THEN
        VERIFY pStatusFlags = (FALSE, FALSE, ?, ?)
    VERIFY pCurrentState = NORMAL
    IF (Protocol_Revision is present AND Protocol_Revision >= 1) THEN
        VERIFY Event_Time_Stamps = (*,*,TS2)
```

Notes to Tester: The time stamps indicated by "" can have any valid value.*

[In BTL Specified Tests, delete the following tests. The only change was to the Purpose statement and the BTL-WG voted to drop our change and keep the SSPC version of the test]

Delete 8.5.X1
Delete 8.5.X2
Delete 8.5.X3
Delete 8.5.X4
Delete 8.5.X7
Delete 8.5.X8

[In BTL Specified Tests, renumber test 8.5.X9 to 8.5.17.1 and replace it with the following which is derived from the 135.1-2019 version.]

8.5.17.1 CHANGE_OF_RELIABILITY with the *No* ~~NONE~~ ~~fault~~*Fault* Algorithm (UnconfirmedEventNotifications)

Reason for Change: Changed title to use No Fault vs NONE algorithm. Fixed Status_Flags in steps 4 and 8.

Purpose: ~~To verify the correct operation of the NONE fault algorithm.~~ *To verify the correct operation of an object that supports first stage reliability evaluation and does not apply a standardized fault algorithm.*

Test Concept: Select an object, O1 that supports first stage reliability evaluation and does not apply a standardized fault algorithm. Ensure that no other fault conditions exist for the object. Create a fault condition. Verify the transition to fault is generated with Reliability set to R1. Remove the fault condition and verify the object transitions out of fault.

Configuration Requirements: O1 is configured to detect and report faults, to have no fault conditions present and the 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(O1 enter a fault condition)
4. 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' =	CHANGE_OF_RELIABILITY,
'Message Text' =	(optional, any valid message text),
'Notify Type' =	ALARM | EVENT,
'AckRequired' =	TRUE | FALSE,
'From State' =	NORMAL,
'To State' =	FAULT,
'Event Values' =	(R1 any valid BACnetReliability, (~~?~~*T*, T, ?, ?), (A list of valid values for properties required to be reported for O1, and 0 or more other properties of O1))

5. VERIFY pCurrentReliability = R1
6. VERIFY pCurrentState = FAULT
7. MAKE(O1clear the fault condition)
8. 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' =	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,
                                          (~~?~~*F*, F, ?, ?),
                                          (A list of valid values for properties required to be reported
                                          for O1, and 0 or more other properties of O1)
                                          )
9.  VERIFY pCurrentReliability = NO_FAULT_DETECTED
10. VERIFY pCurrentState = NORMAL
```

Notes to Tester: The mechanism to enter the *no fault state* ~~NONE fault algorithm~~ is a local matter.

[In BTL Specified Tests, change test 8.5.X9.2 to 8.5.17.2 and then replace the test with the below version which is derived from 135.1-2019.]

8.5.17.2 CHANGE_OF_RELIABILITY with the FAULT_CHARACTERSTRING Algorithm (UnconfirmedEventNotifications)

Reason for Change: Added steps to verify the pCurrentState after each transition change.
Purpose: To verify the correct operation of the FAULT_CHARACTERSTRING fault algorithm.

Test Concept: Select a fault detecting object O1 which is configured to use the FAULT_CHARACTERSTRING algorithm, and no other fault conditions exist for the object. pMonitoredValue is changed to a fault string and back to a non-fault string. It is verified that O1 generates the correct transitions.

Configuration Requirements: O1 is configured to detect and report faults, to have no fault conditions present, and to be in the NORMAL state. FVSET is the set of character strings defined as fault values for O1. ONVSET is the set of character strings defined as offnormal values for O1. FV1 contain a substring that exists in FVSET. If the empty string is included in the FVSET, then FV1 should be the empty string. NFV1 is a string value that does not contain substrings from FVSET or ONVSET. The 'Issue Confirmed Notifications' parameter shall have a value of FALSE.

Test Steps:

```
1.  VERIFY pCurrentReliability = NO_FAULT_DETECTED
2.  VERIFY pCurrentState = NORMAL
3.  IF (pMonitoredValue is writable) THEN
        WRITE pMonitoredValue = FV1
    ELSE
        MAKE (pMonitoredValue = FV1)
4.  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' =                    CHANGE_OF_RELIABILITY,
            'Message Text' =                  (optional, any valid message text),
            'Notify Type' =                   ALARM | EVENT,
            'AckRequired' =                   TRUE | FALSE,
            'From State' =                    NORMAL,
            'To State' =                      FAULT,
            'Event Values' =                  (MULTI_STATE_FAULT,
                                              (T, T, ?, ?),
                                              (A list of valid values for properties required to be reported
                                               for O1, and 0 or more other properties of O1)
                                              )
```

```
5.  VERIFY pCurrentReliability = MULTI_STATE_FAULT
6.  VERIFY pCurrentState = FAULT
7~~6~~. IF (pMonitoredValue is writable) THEN
        WRITE pMonitoredValue = NFV1
    ELSE
        MAKE (pMonitoredValue = NFV1)
8~~7~~. 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' =                     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,
                                                (F, F, ?, ?),
                                                (A list of valid values for properties required to be reported
                                                 for O1, and 0 or more other properties of O1)
                                               )
9.  VERIFY pCurrentReliability = NO_FAULT_DETECTED
10. VERIFY pCurrentState = NORMAL
```

Notes to Tester: Note that a string is considered a substring of itself. Values required and allowed for O1 are described in standard 135 as "Properties Reported in CHANGE_OF_RELIABILITY Notifications" (Table 13-5 in 135-2016) along with supporting paragraphs.

[In BTL Specified Tests, renumber test 8.5.X9.3 to 8.5.17.3 and then replace with the following test which is derived from 135.1-2019.]

8.5.17.3 CHANGE_OF_RELIABILITY with the FAULT_EXTENDED Algorithm (UnconfirmedEventNotification)

Reason for Change: Added missing verification of pCurrentState after each transition and fixed State_Flags in step 8.

Purpose: To verify the correct operation of the FAULT_EXTENDED fault algorithm.

Test Concept: Select a fault detecting object, O1, which is configured to use the FAULT_EXTENDED algorithm, and either pMonitoredValue is configured. Ensure that no other fault conditions exist for the object. In object O1, a condition is created that is detected as a fault by the FAULT_EXTENDED algorithm configured. The fault condition is then removed. It is verified that O1 generates the correct notifications.

Configuration Requirements: O1 is configured to detect and report faults, to have no fault conditions present, and has an Event_State of 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 (a fault condition exist)
4.  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' =	CHANGE_OF_RELIABILITY,
'Message Text' =	(optional, any valid message text),
'Notify Type' =	ALARM | EVENT,
'AckRequired' =	TRUE | FALSE,
'From State' =	NORMAL,
'To State' =	FAULT,
'Event Values' =	((R1: any valid reliability value), (T, T, ?, ?), (a vendor specified set of values))

5. VERIFY pCurrentReliability = R1
6. *VERIFY pCurrentState = FAULT*
7. MAKE (remove the fault condition)
8. 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' =	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, (~~?~~*F*, F, ?, ?), (a vendor specified set of values))

9. VERIFY pCurrentReliability = NO_FAULT_DETECTED
10. *VERIFY pCurrentState = NORMAL*

[In BTL Specified Tests, renumber test 8.5.X9.4 to 8.5.17.4 and then replace with the following test which is derived from 135.1-2019.]

8.5.17.4 CHANGE_OF_RELIABILITY with the FAULT_LIFE_SAFETY Algorithm (UnconfirmedEventNotifications)

Reason for Change: Added verification of pCurrentState after each transition.

Purpose: To verify the correct operation of the FAULT_LIFE_SAFETY fault algorithm.

Test Concept: Select a fault detecting object, O1, which is configured to use the FAULT_LIFE_SAFETY algorithm. Ensure that no other fault conditions exist in the object. Set pMonitoredValue to FV1, a value which indicates a FAULT_LIFE_SAFETY fault condition. Verify the correct transition is generated. The fault condition is removed by setting pMonitoredValue to NV1, a value which indicates NO_FAULT_DETECTED and verify the correct transition is generated.

Configuration Requirements: O1 is configured to detect. O1 is initially configured to have no fault conditions present, and has an Event_State of NORMAL. FV1 is a value for pMonitoredValue which indicates a fault condition, and NV1 is a value for pMonitoredValue which does not indicate a fault condition. The 'Issue Confirmed Notifications' parameter shall have a value of FALSE.

Test Steps:
1. VERIFY pCurrentReliability = NO_FAULT_DETECTED
2. VERIFY pCurrentState = NORMAL
3. IF (pMonitoredValue is writable) THEN

```
        WRITE pMonitoredValue = FV1
    ELSE
        MAKE (pMonitoredValue = FV1)
4.  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' =                    CHANGE_OF_RELIABILITY,
            'Message Text' =                  (optional, any valid message text),
            'Notify Type' =                   ALARM | EVENT,
            'AckRequired' =                   TRUE | FALSE,
            'From State' =                    NORMAL,
            'To State' =                      FAULT,
            'Event Values' =                  (MULTI_STATE_FAULT,
                                               (T, T, ?, ?),
                                               (A list of valid values for properties required to be reported
                                                for O1, and 0 or more other properties of O1)
                                              )
5.  VERIFY pCurrentReliability = MULTI_STATE_FAULT
6.  VERIFY pCurrentState = FAULT
7~~6~~. IF (pMonitoredValue is writable) THEN
        WRITE pMonitoredValue = NV1
    ELSE
        MAKE (pMonitoredValue = NV1)
8~~7~~. 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' =                    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,
                                               (F, F, ?, ?),
                                               (A list of valid values for properties required to be reported
                                                for O1, and 0 or more other properties of O1)
                                              )
9~~8~~. VERIFY pCurrentReliability = NO_FAULT_DETECTED
10. VERIFY pCurrentState = NORMAL
```

[In BTL Specified Tests, renumber test 8.5.X9.5 to 8.5.17.5 and then replace with the following version derived from 135.1-2019.]

8.5.17.5 CHANGE_OF_RELIABILITY with the FAULT_STATE Algorithm (UnconfirmedEventNotifications)

Reason for Change: Added verification of pCurrentState after each transition.

Purpose: To verify the correct operation of the FAULT_STATE fault algorithm.

Test Concept: Select a fault detecting object, O1, which is configured to use the FAULT_STATE algorithm. Ensure that no other fault conditions exist in the object. Set pMonitoredValue to FV1, a value which indicates a FAULT_STATE fault condition. Verify the correct transition is generated. The fault condition is removed by setting pMonitoredValue to NV1, a value which indicates NO_FAULT_DETECTED and verify the correct transition is generated.

Configuration Requirements: O1 is configured to detect faults. O1 is initially configured to have no fault conditions present, and an Event_State of NORMAL. FV1 is a value for pMonitoredValue which indicates a fault condition, and NV1 is a value for pMonitoredValue which does not indicate a fault condition. The 'Issue Confirmed Notifications' parameter shall have a value of FALSE.

Test Steps:

```
1.  VERIFY pCurrentReliability = NO_FAULT_DETECTED
2.  VERIFY pCurrentState = NORMAL
3.  IF (pMonitoredValue is writable) THEN
        WRITE pMonitoredValue = FV1
    ELSE
        MAKE (pMonitoredValue = FV1)
4.  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' =                    CHANGE_OF_RELIABILITY,
            'Message Text' =                  (optional, any valid message text),
            'Notify Type' =                   ALARM | EVENT,
            'AckRequired' =                   TRUE | FALSE,
            'From State' =                    NORMAL,
            'To State' =                      FAULT,
            'Event Values' =                  (MULTI_STATE_FAULT,
                                               (T, T, ?, ?),
                                                (A list of valid values for properties required to be reported
                                                for O1, and 0 or more other properties of O1)
                                              )
5.  VERIFY pCurrentReliability = MULTI_STATE_FAULT
6.  VERIFY pCurrentState = FAULT
7~~6~~. IF (pMonitoredValue is writable) THEN
        WRITE pMonitoredValue = NV1
    ELSE
        MAKE (pMonitoredValue = NV1)
8~~7~~. 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' =                    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,
                                               (F, F, ?, ?),
                                               (A list of valid values for properties required to be reported
                                               for O1, and 0 or more other properties of O1)
```

```
                                        )
9.  VERIFY pCurrentReliability = NO_FAULT_DETECTED
*10. VERIFY pCurrentState = NORMAL*
```

[In BTL Specified Tests, renumber test 8.5.X9.6 to 8.5.17.6 and then replace with the following test which is derived from 135.1-2019.]

8.5.17.6 CHANGE_OF_RELIABILITY with the FAULT_STATUS_FLAGS Algorithm (UnconfirmedEventNotifications)

Reason for Change: Added verification of pCurrentState after each transition.

Purpose: To verify the correct operation of the FAULT_STATUS_FLAGS fault algorithm.

Test Concept: Select a fault detecting object, O1, which is configured to use the FAULT_STATUS_FLAGS algorithm. Ensure that no other fault conditions exist for the object. Set pMonitoredValue to FV1, a value which indicates a FAULT_STATUS_FLAGS fault condition. Verify the correct transition is generated. The fault condition is removed by setting pMonitoredValue to NV1, a value which indicates NO_FAULT_DETECTED and verify the correct transition is generated.

Configuration Requirements: O1 is configured to detect. O1 is initially configured to have no fault conditions present, and Event_State is NORMAL. FV1 is a value for pMonitoredValue which indicates a fault condition, and NV1 is a value for pMonitoredValue which does not indicate a fault condition. The 'Issue Confirmed Notifications' parameter shall have a value of FALSE.

Test Steps:

```
1.  VERIFY pCurrentReliability = NO_FAULT_DETECTED
2.  VERIFY pCurrentState = NORMAL
3.  IF (pMonitoredValue is writable) THEN
        WRITE pMonitoredValue = FV1
    ELSE
        MAKE (pMonitoredValue = FV1)
4.  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' =                    CHANGE_OF_RELIABILITY,
            'Message Text' =                  (optional, any valid message text),
            'Notify Type' =                   ALARM | EVENT,
            'AckRequired' =                   TRUE | FALSE,
            'From State' =                    NORMAL,
            'To State' =                      FAULT,
            'Event Values' =                  (MEMBER_FAULT,
                                               (T, T, ?, ?),
                                                (A list of valid values for properties required to be reported
                                                for O1, and 0 or more other properties of O1)
                                              )
5.  VERIFY pCurrentReliability = MEMBER_FAULT
*6.  VERIFY pCurrentState = FAULT*
7~~6~~. IF (pMonitoredValue is writable) THEN
        WRITE pMonitoredValue = NV1
    ELSE
        MAKE (pMonitoredValue = NV1)
8~~7~~. 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' =	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,
	(F, F, ?, ?),
	(A list of valid values for properties required to be reported for O1, and 0 or more other properties of O1)
	)

9~~8~~. VERIFY pCurrentReliability = NO_FAULT_DETECTED
10. VERIFY pCurrentState = NORMAL

[In BTL Specified Tests, Renumber 8.5.X9.8 to 8.5.17.5 and then replace with version below derived from 135.1-2019.]

8.5.17.8 CHANGE_OF_RELIABILITY of Event Enrollment Object, Monitored Object Fault (UnconfirmedEventNotifications)

Reason for Change: Added verify of pCurrentState after each transition.

Purpose: To verify the proper operation of the Event Enrollment object's fault detection when the monitored object enters the fault state.

Test Concept: Select an Event Enrollment object, EE1, that monitors an object, M1, that can transition into FAULT. Starting with both objects in a NORMAL state, cause a condition which results in a fault in M1. Verify EE1 reports the fault. Clear the condition and verify EE1 reports the return to NORMAL.

Configuration Requirements: EE1 is configured to process faults in M1. EE1 and M1 are each initially 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 (M1 enter any fault state)
4. BEFORE **Notification Fail Time**
    RECEIVE UnconfirmedEventNotification-Request

'Process Identifier' =	(any valid process identifier),
'Initiating Device Identifier' =	IUT,
'Event Object Identifier' =	EE1,
'Time Stamp' =	(any valid time stamp),
'Notification Class' =	(the notification class configured for EE1),
'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' =	NORMAL,
'To State' =	FAULT,
'Event Values' =	(MONITORED_OBJECT_FAULT,
	(T, T, ?, ?),
	M1,
	(optional, property value of M1),

```
                                        (optional, M1 Status_Flags, (?, T, ?, ?)),  Should be (?,T,?,?)
                                        (0 or more other properties of M1)
                                       )
```

5. VERIFY pCurrentReliability = MONITORED_OBJECT_FAULT
6. VERIFY pCurrentState = FAULT
7. MAKE (M1 clear fault state)
8. BEFORE **Notification Fail Time**

```
      RECEIVE UnconfirmedEventNotification-Request
            'Process Identifier' =              (any valid process identifier),
            'Initiating Device Identifier' =    IUT,
            'Event Object Identifier' =         EE1,
            'Time Stamp' =                      (any valid time stamp),
            'Notification Class' =              (the notification class configured for EE1),
            '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,
                                                 (F, F, ?, ?),
                                                  M1,
                                                  (optional, property value of M1),
                                                  (optional, M1 Status_Flags, (?, F, ?, ?)),
                                                  (0 or more other properties of M1)
                                                )
```

9. VERIFY pCurrentReliability = NO_FAULT_DETECTED
10. *VERIFY pCurrentState = NORMAL*

[In BTL Specified Tests, renumber test 8.5.X9.9 to 8.5.17.9 and then replace with the following test as derived from 135.1-2019.]

8.5.17.9 CHANGE_OF_RELIABILITY of Event Enrollment Object Fault (UnconfirmedEventNotifications)

Reason for Change: Improved Configuration Requirements text, added check of state at end of test.

Purpose: To verify the Event Enrollment object generates a fault event when the object enters ~~into~~ fault due to an internal unreliable operation.

Test Concept: Select an Event Enrollment object EE1 that can be made to enter into fault due to an internal unreliable operation. Starting EE1 in a NORMAL state, cause a condition which results in an internal fault. Verify that EE1 reports the fault. Clear the condition and verify that EE1 reports the return to NORMAL.

Configuration Requirements: EE1 is configured to be able to enter a fault state and to report those *using unconfirmed event notifications.* EE1 is initially 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 (EE1 enter any internal fault state)
4. BEFORE **Notification Fail Time**

```
      RECEIVE UnconfirmedEventNotification-Request
            'Process Identifier' =              (any valid process identifier),
            'Initiating Device Identifier' =    IUT,
            'Event Object Identifier' =         EE1,
            'Time Stamp' =                      (any valid time stamp),
            'Notification Class' =              (the notification class configured for EE1),
            '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' =                  NORMAL,
            'To State' =                    FAULT,
            'Event Values' =                ((R1: any value other than
                                                    MONITORED_OBJECT_FAULT
                                                    and NO_FAULT_DETECTED),
                                              (T, T, ?, ?),
                                              (M1, any valid monitored object),
                                              (optional, property value of M1),
                                              (optional, M1 Status_Flags, (?, F, ?, ?)),
                                              (0 or more other properties of M1)
                                            )
5.  VERIFY pCurrentReliability = R1
6.  VERIFY pCurrentState = FAULT
7.  MAKE (EE1 clear fault state)
8.  BEFORE **Notification Fail Time**
        RECEIVE UnconfirmedEventNotification-Request
            'Process Identifier' =          (any valid process identifier),
            'Initiating Device Identifier' = IUT,
            'Event Object Identifier' =     EE1,
            'Time Stamp' =                  (any valid time stamp),
            'Notification Class' =          (the notification class configured for EE1),
            '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,
                                              (F, F, ?, ?),
                                              M1,
                                              (optional, property value of M1),
                                              (optional, M1 Status_Flags, (?, F, ?, ?)),
                                              (0 or more other properties of M1)
                                            )
9.  VERIFY pCurrentReliability = NO_FAULT_DETECTED
10. VERIFY pCurrentState = NORMAL
```

[In BTL Specified Tests, renumber test 8.5.X9.10 to 8.5.17.10 and then replace with copy below derived from 135.1-2019.]

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

Reason for Change: Fix the Status_Flags in step 7. Add verify steps 12 and 13.

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 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.  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)
5.  VERIFY pCurrentState = OFFNORMAL
6.  MAKE(O1 enter a fault state)
7.  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' =                    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),
                                                (~~?~~T, T, ?, ?),
                                                (A list of valid values for properties required to be reported
                                                for O1, and 0 or more other properties of O1)
                                              )
8.  MAKE(O1 clear the fault condition)
9.  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' =                    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,
                                                (F, F, ?, ?),
                                                (A list of valid values for properties required to be reported
                                                for O1, and 0 or more other properties of O1)
```

)

10. VERIFY pCurrentReliability = NO_FAULT_DETECTED
11. 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,
    '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)
12. *VERIFY pCurrentReliability = NO_FAULT_DETECTED*
13. *VERIFY pCurrentState = OFFNORMAL*

[In BTL Specified Tests, update this test with changes highlighted below.]

8.5.X9.11 CHANGE_OF_RELIABILITY with First Stage Object Fault (UnconfirmedEventNotification)

Reason for Change: Fixed status_flags value in request.

Purpose: To verify that fault conditions due to first stage faults are detected and reported.

Test Concept: An object in the IUT, O1, which can detect at least one first stage fault is selected. One of O1's detectable first stage faults, R1, is selected for the test. O1 begins the test in the NORMAL state with pCurrentReliability equal to NO_FAULT_DETECTED. The first stage fault condition, R1, is made to exist and it is verified that the pCurrentReliability changes to R1. It is verified that O1 generates the appropriate event notification. The fault condition is removed, and it is verified that the pCurrentReliability returns to NO_FAULT_DETECTED and the appropriate event notification message is generated.

Configuration Requirements: O1 is configured to detect faults and to report those using unconfirmed event notifications. O1 is initially configured to have no fault conditions present, and Event_State is NORMAL.

Test Steps:

1. VERIFY pCurrentReliability = NO_FAULT_DETECTED
2. VERIFY pCurrentState = NORMAL
3. MAKE (pCurrentReliability = R1)
4. BEFORE **Notification Fail Time**
    RECEIVE UnconfirmedEventNotification-Request,
    'Process Identifier' = (any valid process ID),
    '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' = CHANGE_OF_RELIABILITY,
    'Message Text' = (optional, any valid message text),
    'Notify Type' = EVENT | ALARM,
    'AckRequired' = TRUE | FALSE,
    'From State' = NORMAL,
    'To State' = FAULT,
    'Event Values' = (R1,
    (~~?~~*T*, T, ?, ?),
    (A list of valid values for properties required to be reported for O1, and 0 or more other properties of O1)

```
                                        )
5.  VERIFY pCurrentReliability = R1
6.  VERIFY pCurrentState = FAULT
7.  MAKE (pCurrentReliability = NO_FAULT_DETECTED)
8.  BEFORE Notification Fail Time
        RECEIVE UnconfirmedEventNotification-Request,
            'Process Identifier' =              (any valid process ID),
            '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' =                     EVENT | ALARM,
            'AckRequired' =                     TRUE | FALSE,
            'From State' =                      FAULT,
            'To State' =                        NORMAL,
            'Event Values' =                    (NO_FAULT_DETECTED,
                                                 (~~?~~*F*, F, ?, ?),
                                                 (A list of valid values for properties required to be reported
                                                  for O1, and 0 or more other properties of O1)
                                                )
9.  VERIFY pCurrentReliability = NO_FAULT_DETECTED
10. VERIFY pCurrentState = NORMAL
```

[In BTL Specified Tests, replace test 9.1.1.1 to derive from 135.1-2019 as shown below.]

9.1.1.1 Successful Alarm Acknowledgment of Confirmed Event Notifications Using the Time Form of the 'Time of Acknowledgment' Parameter

Reason for Change: Made changes to allow cases where only one Recipient_List entry is supported.

Purpose: To verify the successful acknowledgment of an alarm signaled by a ConfirmedEventNotification, including notification of other workstations and updating of the Acked_Transitions status. The Time form of the 'Time of Acknowledgment' parameter is used.

Test Concept: An alarm is triggered that causes the IUT to notify the TD and ~~one other device~~ *all other recipients in the Recipient_List*. The TD acknowledges the alarm and verifies that the acknowledgment is properly noted by the IUT. The IUT notifies all other recipients that the alarm has been acknowledged.

Configuration Requirements: The IUT shall be configured with ~~at~~ *an* object that can detect alarm conditions and send confirmed notifications. The Acked_Transitions property shall have the value B'111' indicating that all transitions have been acknowledged. The TD and one other BACnet device, if the IUT supports multiple recipients, shall be recipients of the alarm notification.

Test Steps:

```
1.  MAKE (a change that triggers the detection of an alarm event in the IUT)
2.  WAIT (pTimeDelay)
3.  BEFORE Notification Fail Time
        RECEIVE ConfirmedEventNotification-Request,
            'Process Identifier' =              (the process identifier configured for this event),
            'Initiating Device Identifier' =    IUT,
            'Event Object Identifier' =         (the object detecting the alarm),
            'Time Stamp' =                      (T1: any valid time stamp),
            'Notification Class' =              (the notification class configured for this event),
            'Priority' =                        (the priority configured for this event),
            'Event Type' =                      (any valid event type),
            'Message Text' =                    (optional, any valid message text),
            'Notify Type' =                     (the notify type configured for this event),
            'AckRequired' =                   TRUE,
            'From State' =                    NORMAL,
            'To State' =                      (any appropriate non-normal event state),
            'Event Values' =                  (the values appropriate to the event type)
4.  TRANSMIT BACnet-SimpleACK-PDU
5.  RECEIVE
        DESTINATION =                         (a device other than the TD),
        SOURCE =                              IUT,
        ConfirmedEventNotification-Request,
        'Process Identifier' =                (the process identifier configured for this event),
        'Initiating Device Identifier' =      IUT,
        'Event Object Identifier' =           (the object detecting the alarm),
        'Time Stamp' =                        (T1: any valid time),
        'Notification Class' =                (the notification class configured for this event),
        'Priority' =                          (the priority configured for this event),
        'Event Type' =                        (any valid event type),
        'Message Text' =                      (optional, any valid message text),
        'Notify Type' =                       (the notify type configured for this event),
        'AckRequired' =                       TRUE,
        'From State' =                        NORMAL,
        'To State' =                          (any appropriate non-normal event state),
        'Event Values' =                      (the values appropriate to the event type)
6.  TRANSMIT BACnet-SimpleACK-PDU
7.  VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = (FALSE, TRUE, TRUE)
8.  TRANSMIT AcknowledgeAlarm-Request,
        'Acknowledging Process Identifier' =  (the value of the 'Process Identifier' parameter in the event
                                              notification),
        'Event Object Identifier' =           (the 'Event Object Identifier' from the event notification),
        'Event State Acknowledged' =          (the state specified in the 'To State' parameter of the notification),
        'Time Stamp' =                        (the time stamp conveyed in the notification),
        'Time of Acknowledgment' =            (the TD's current time using a Time format)
9.  RECEIVE BACnet-Simple-ACK-PDU
10. IF (Protocol_Revision is present and Protocol_Revision ≥ 1) THEN
        BEFORE **Notification Fail Time**
            RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' =            (the process identifier configured for this event),
                'Initiating Device Identifier' =  IUT,
                'Event Object Identifier' =       (the object detecting the alarm),
                'Time Stamp' =                    (any valid time stamp),
                'Notification Class' =            (the notification class configured for this event),
                'Priority' =                      (the priority configured for this event),
                'Event Type' =                    (the event type included in step 3),
                'Message Text' =                  (optional, any valid message text),
                'Notify Type' =                   ACK_NOTIFICATION,
                'To State' =                      (the 'To State' used in step 3)
    ELSE
        BEFORE **Notification Fail Time**
            RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' =            (the process identifier configured for this event),
                'Initiating Device Identifier' =  IUT,
                'Event Object Identifier' =       (the object detecting the alarm),
                'Time Stamp' =                    (any valid time stamp),
                'Notification Class' =            (the notification class configured for this event),
                'Priority' =                      (the priority configured for this event),
                'Event Type' =                    (the event type included in step 3),
                'Message Text' =                  (optional, any valid message text),
                'Notify Type' =                   ACK_NOTIFICATION,
                ~~'To State' =                      (the 'To State' used in step 3)~~
11. TRANSMIT BACnet-SimpleACK-PDU
12. IF (Protocol_Revision is present and Protocol_Revision ≥ 1) THEN
```

```
        BEFORE Notification Fail Time
            RECEIVE
                DESTINATION =                    (a device other than the TD),
                SOURCE =                         IUT,
                ConfirmedEventNotification-Request,
                'Process Identifier' =           (the process identifier configured for this event),
                'Initiating Device Identifier' = IUT,
                'Event Object Identifier' =      (the object detecting the alarm),
                'Time Stamp' =                   (the timestamp or sequence number received in step 10),
                'Notification Class' =           (the notification class configured for this event),
                'Priority' =                     (the priority configured for this event),
                'Event Type' =                   (the event type included in step 3),
                'Message Text' =                 (optional, any valid message text),
                'Notify Type' =                  ACK_NOTIFICATION,
                'To State' =                     (the 'To State' used in step 3)
    ELSE
        BEFORE Notification Fail Time
            RECEIVE
                DESTINATION =                    (at least one device other than the TD),
                SOURCE =                         IUT,
                ConfirmedEventNotification-Request,
                'Process Identifier' =           (the process identifier configured for this event),
                'Initiating Device Identifier' = IUT,
                'Event Object Identifier' =      (the object detecting the alarm),
                'Time Stamp' =                   (the timestamp or sequence number received in step 10),
                'Notification Class' =           (the notification class configured for this event),
                'Priority' =                     (the priority configured for this event),
                'Event Type' =                   (the event type included in step 3),
                'Message Text' =                 (optional, any valid message text),
                'Notify Type' =                  ACK_NOTIFICATION
13. TRANSMIT BACnet-SimpleACK-PDU
14. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = (TRUE,TRUE,TRUE)
```

Notes to Tester: The destination address used for the acknowledgment notification in step 12 shall be the same address used in step 5. Inclusion of the 'To State' parameter in acknowledgement notifications was added in protocol version 1, protocol revision 1. Implementations that precede this version will not include this parameter. When multiple event notifications are expected for a specific event, the order that the IUT transmits them in is irrelevant. *If the IUT can only be configured with one recipient in the Recipient_List property of the issuing Notification_class object, skip all steps related to receipt of the second notification.*

[In BTL Specified Tests, delete the following tests.]
Delete 9.1.1.2
Delete 9.1.1.3
Delete 9.1.1.4
Delete 9.1.1.5
Delete 9.1.1.6
Delete 9.1.1.8
Delete 9.1.1.9
Delete 9.1.1.10
Delete 9.1.1.11
Delete 9.1.1.X3
Delete 9.1.X1

Delete 9.1.2.3
Delete 9.1.2.4
Delete 9.1.2.5
Delete 9.1.2.6
Delete 9.1.2.7

Delete 9.4.5

Delete 9.4.6
Delete 9.4.X1

Delete 9.5.X1

Delete 9.7.1.1
Delete 9.7.2.3

Delete 9.8.6

[In BTL Specified Tests, update test 9.11.2.1 to derive from 135.1-2019 as shown below.]

9.11.2.1 The Monitored Object Does Not Support COV Notification

Reason for Change: Added additional acceptable error responses.

Purpose: To verify that the IUT correctly responds to a SubscribeCOVProperty request to establish a subscription when the monitored object does not support COV notifications.

Test Steps:

```
1.  TRANSMIT SubscribeCOVProperty-Request,
        'Subscriber Process Identifier' =     (any valid process identifier),
        'Monitored Object Identifier' =       (any object that does not support COV notifications),
        'Issue Confirmed Notifications' =     TRUE,
        'Lifetime' =   60,
        'Monitored Property Identifier' =    (any property in the object)
2.  IF (Protocol_Revision < 15) THEN
        RECEIVE
            (BACnet-Error-PDU,
                Error Class =      SERVICES,
                Error Code =       SERVICE_REQUEST_DENIED | OTHER) |
            (BACnet-Error-PDU,
                Error Class =      OBJECT,
                Error Code =       OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED) |
            (BACnet-Error-PDU,
                Error Class =      PROPERTY,
                Error Code =       NOT_COV_PROPERTY)
    ELSE
        RECEIVE
            (BACnet-Error-PDU,
                Error Class =      OBJECT,
                Error Code =       OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED) |
            (BACnet-Error-PDU,
                Error Class =      PROPERTY,
                Error Code =       NOT_COV_PROPERTY)
```

9.11.2.2 The Monitored Property Does Not Support COV Notification

Reason for Change: Changed description of Monitored Property Identifier to be chosen for this test.

Purpose: To verify that the IUT correctly responds to a SubscribeCOVProperty request to establish a subscription when the monitored object supports COV notifications but not on the requested property.

Test Steps:

```
1.  TRANSMIT SubscribeCOVProperty-Request,
        'Subscriber Process Identifier' =     (any valid process identifier),
        'Monitored Object Identifier' =       (any object that supports COV notifications),
        'Issue Confirmed Notifications' =     TRUE,
        'Lifetime' =                          60,
        'Monitored Property Identifier' =     (any property of the chosen object that does not support COV notifications)
2.  IF (Protocol_Revision < 15) THEN
        RECEIVE
            (BACnet-Error-PDU,
                Error Class =     SERVICES,
                Error Code =      SERVICE_REQUEST_DENIED | OTHER) |
            (BACnet-Error-PDU,
                Error Class =     PROPERTY,
                Error Code =      NOT_COV_PROPERTY)
    ELSE
        RECEIVE BACnet-Error-PDU,
            Error Class =     PROPERTY,
            Error Code =      NOT_COV_PROPERTY
```

[In BTL Specified Tests, delete the following tests.]
Delete 9.16.2.1
Delete 9.16.2.6
Delete test 9.20.1.7
Delete test 9.20.1.9
Delete 9.31.1.2
Delete 9.33.1.3

[In BTL Specified Tests, delete text for Section 10.8 (now in the standard and not needed here)]
[In BTL Specified Tests, remove figure 10.8.X1 (in standard as 10-4)]
[In BTL Specified Tests, remove section text under 10.8.7 Multiple Devices on a Single Virtual Network. This exists already in the standard.]

[In BTL Specified Tests, renumber section 13.X13 General Application State Machine Tests to 13.9 Application State Machine Tests.]
[In BTL Specified Tests, renumber test 13.X13.1 to 13.9.X1]

[In BTL Specified Tests, change title of 14.7 from Broadcast management … to Broadcast Management …]

Errata for 135.1-2019:

In 135,1-2019 – test 7.3.1.11 has e instead of >= in step 3 and step 10. Step 18 has italic and should not, has ‘acknowledgement-source’ and ‘timestamp’ in the wrong order, and has misplaced quote around ‘to-state’.

In 135.1-2019 -- test 7.3.1.17 has italic in it.
In 135.1-2019 -- test 7.3.2.9.7, step 3 has e instead of >=.

In 135.1-2019 -- test 7.3.2.31.2 has a strike through on step 1.

In 135.1-2019 Test 8.4.9 has e instead of >= in two places. This test also does not have test step numbers they are all under step 1.

In 135.1-2019 Test 8.5.17.9 – Configuration Requirements has an incomplete sentence.

In 135.1-2019 Test 8.5.17.10 -- test steps are messed up.

In 135.1-2019 Test 9.1.1.1 – uses the word ‘at’ instead of the word ‘an’ in Configuration Requirements.

In 135.1-2019 test 13.8.2.1 under Backup Characteristics #8, has e instead of >=. Same under Restore Characteristics #8 has e instead of >=.