[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.

BTL-TP14.0e-1: Add Support for Accumulator Object, pg 2. [wID0061]

In the following document, language to be added to existing clauses within the BTL Test Package 14.0 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 addition, changes to BTL Specified Tests might 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.

This addendum contains results of various clarification requests put forth to the BTL-WG that resulted in test package changes.

BTL-TP14.0e-1: Add Support for Accumulator Object

Overview:

This document applies changes to the Checklist, Test Plan and Specified Tests for:

- Add Accumulator object type.

Changes:

[In BTL Checklist, add new Accumulator section in existing 3. Object testing]

Accumulator		
	R	Base Requirements
	O	Supports Prescale property
	O	Supports Logging_Record property
	O	Supports writable Out_Of_Service properties
	O	Supports writable Value_Set property
	O	Supports writable Value_Before_Change property

[In BTL Checklist, add new algorithm section in existing 5.2 AE-N-I-B and 5.3 AE-N-E-B testing]

Alarm and Event - Notification - Internal - B		
	R	Base Requirements
	R	Supports AE-INFO-B
	R	Supports the Notification Class object
	C[1]	Supports AE-ACK-B
	C[2]	Supports intrinsic alarming
	C[2]	Supports the Event Enrollment object
	C[3]	Implements the CHANGE_OF_BITSTRING algorithm
	C[3]	Implements the CHANGE_OF_STATE algorithm
	C[3]	Implements the numeric form of the CHANGE_OF_VALUE algorithm
	C[3]	Implements the Bit String form of the CHANGE_OF_VALUE algorithm
	C[3]	Implements the COMMAND_FAILURE algorithm
	C[3]	Implements the FLOATING_LIMIT algorithm
	C[3]	Implements the OUT_OF_RANGE algorithm
	C[3]	Implements a Proprietary algorithm using Complex notifications
	C[3]	Implements a Proprietary algorithm using Extended notifications
	C[4]	Generates event notifications with timestamps of the BACnetDateTime form
	C[4]	Generates event notifications with timestamps of the Time form
	C[4]	Generates event notifications with timestamps of the Sequence Number f orm
	O	Implements intrinsic alarming in an Analog object
	O	Supports writable Event_Parameters properties
	C[3]	Implements the DOUBLE_OUT_OF_RANGE algorithm
	C[3]	Implements the SIGNED_OUT_OF_RANGE algorithm
	C[3]	Implements the UNSIGNED_OUT_OF_RANGE algorithm
	C[3]	Implements the CHANGE_OF_CHARACTERSTRING algorithm
	C[3]	Implements the UNSIGNED_RANGE algorithm
	O	Supports Event_Message_Texts property
	O	Supports Event_Message_Texts_Config property
		[1] Required if EventNotifications with service parameter AckRequired = True can be issued. [2] At least one of these options must be supported to claim support for this BIBB. [3] At least one of these options must be supported to claim support for this BIBB. It is recommended that a standard BACnet algorithm be used instead of a proprietary algorithm whenever possible. [4] At least one of these options must be supported to claim support for this BIBB. The BACnetDateTime form of the timestamp is the recommended option.
Alarm and Event - Notification - External - B		
	R	Base Requirements
	R	Supports AE-N-I-B

	R	Supports DS-RP-A for retrieving monitored values
	R	Supports the Event Enrollment object
	C[1]	Implements the CHANGE_OF_BITSTRING algorithm
	C[1]	Implements the CHANGE_OF_STATE algorithm
	C[1]	Implements the numeric form of the CHANGE_OF_VALUE algorithm
	C[1]	Implements the Bit String form of the CHANGE_OF_VALUE algorithm
	C[1]	Implements the COMMAND_FAILURE algorithm
	C[1]	Implements the FLOATING_LIMIT algorithm
	C[1]	Implements the OUT_OF_RANGE algorithm
	C[1]	Implements the DOUBLE_OUT_OF_RANGE algorithm
	C[1]	Implements the SIGNED_OUT_OF_RANGE algorithm
	C[1]	Implements the UNSIGNED_OUT_OF_RANGE algorithm
	C[1]	Implements the CHANGE_OF_CHARACTERSTRING algorithm
	C[1]	Implements the UNSIGNED_RANGE algorithm
	C[1]	Implements a Proprietary algorithm
	O	Supports Event_Message_Texts property~~.~~
[1] One of these options must be supported to claim support for this BIBB. It is recommended that a standard BACnet algorithm be used instead of a proprietary algorithm whenever possible.		

[In BTL Test Plan, add new Accumulator section at end of existing 3. Object testing, with sections 3.37.1 Base Requirements, and five other 3.37.2 through 3.37.6 sections as indicated.]

• • •

3.37 Accumulator Object

3.37.1 Base Requirements

Base requirements must be met by any IUT that can contain Accumulator objects.

BTL - 7.3.2.X37.1.1 - Present_Value Remains In-Range Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.2.X6 - Datetime Non-Pattern Properties Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision 11 or greater, if the Value_Change_Time property is writable.
	Test Directives	Apply to the Value_Change_Time property, if writable.
	Testing Hints	
	Notes & Results	

3.37.2 Supports Prescale Property

The Prescale property in at least one Accumulator object is present.

BTL - 7.3.2.X37.1.2 - Prescale in Accumulator Test		
	Test Method	Manual

	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.37.3 Supports Logging_Record Property

The Logging_Record property in at least one Accumulator object is present.

BTL - 7.3.2.X37.1.3 - Logging_Record in Accumulator Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.2.X37.1.4 - Logging_Record in Accumulator RECOVERED Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.2.X37.1.5 - Logging_Record in Accumulator STARTING Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.37.4 Supports Writable Out_Of_Service Properties

The Out_Of_Service property in Accumulator objects is writable.

BTL - 7.3.2.X37.1.6 - Out_Of_Service in Accumulator Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.37.5 Supports Writable Value_Set Property

The Value_Set property in Accumulator objects is writable.

BTL - 7.3.2.X37.1.7 - Value_Set Writing Test		
	Test Method	Manual

	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Verify in each object where test 7.3.2.X37.1.7 could be executed, that Value_Before_Change in that object is read-only .
	Testing Hints	
	Notes & Results	

3.37.6 Supports Writable Value_Before_Change Property

The Value_Before_Change property in Accumulator objects is writable.

BTL – 7.3.2.X37.1.8 - Value_Before_Change Writing Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Verify in each object where test 7.3.2.X37.1.8 could be executed, that Value_Set in that object is read-only.
	Testing Hints	
	Notes & Results	

[In BTL Test Plan, add new UNSIGNED_RANGE sections at end of existing algorithms in section 5.2 and at end of existing algorithms before Proprietary in section 5.3 as indicated.]

Typographic conventions: These entirely new sections are shown in normal Test Plan format.

5.2.25 Implements the UNSIGNED_RANGE Algorithm

The IUT contains, or can be made to contain, an object such as an Accumulator object, that can generate EventNotifications with an Event_Type of UNSIGNED_RANGE.

BTL - 8.4.X7 - UNSIGNED_RANGE ConfirmedEventNotification Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	This test must be repeated once for each object type that is capable of generating event notifications with an Event_Type of UNSIGNED_RANGE.
	Testing Hints	
	Notes & Results	
BTL - 8.5.X7 - UNSIGNED_RANGE UnconfirmedEventNotification Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	This test shall be executed with an Event Enrollment object that is configured to monitor a property in a device other than the IUT.
	Testing Hints	
	Notes & Results	

5.3.16 Implements the UNSIGNED_RANGE Algorithm

The IUT contains, or can be made to contain an Event Enrollment object that can generate EventNotifications with an Event_Type of UNSIGNED_RANGE.

BTL - 8.4.X7 - UNSIGNED_RANGE ConfirmedEventNotification Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	This test shall be executed with an Event Enrollment object that is configured to monitor a property in a device other than the IUT.
	Testing Hints	
	Notes & Results	
BTL - 8.5.X7 - UNSIGNED_RANGE UnconfirmedEventNotification Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	This test shall be executed with an Event Enrollment object that is configured to monitor a property in a device other than the IUT.
	Testing Hints	
	Notes & Results	

[In BTL Specified Tests, add new tests 7.3.2.X37.1.1 through 7.3.2.X37.1.5]

7.3.2.X37.1.1 Present_Value Remains In-Range Test

Purpose: To verify the correct wrapping operation of the Accumulator Present_Value.

Test Concept: The IUT shall be configured with a Max_Pres_Value which is attainable, within reasonable testing time, after Present_Value is preset to a value slightly less than that, then incremented. The Present_Value shall remain in range from one to Max_Pres_Value, by wrapping back to 1 when it would exceed Max_Pres_Value.

Test Steps:

```
1.  IF (Value_Set is writable) THEN
        WRITE Value_Set = (a value slightly less than Max_Pres_Value)
    ELSE
        MAKE (Present_Value equal a value slightly less than Max_Pres_Value)
2.  MAKE (the Accumulator increase its Present_Value until it rolls over Max_Pres_Value)
3.  CHECK (Present_Value < Max_Pres_Value)
```

7.3.2.X37.1.2 Prescale in Accumulator Test

Purpose: To verify the correct effect of Prescale on the increment of the Present_Value in Accumulator.

Test Concept: The IUT shall be configured with a Prescale whose effect when incrementing Present_Value is testable. Three readings of the Present_Value are observed, then the math is checked to ensure that it increments at the rate expected given Prescale.

Configuration Requirements: If there is no Prescale property present in any Accumulator object, then this test shall be skipped.

Test Steps:

```
1.  IF (Value_Set is writable) THEN
        WRITE Value_Set = (any valid value $V_1$)
    ELSE
        MAKE (Present_Value equal any valid value $V_1$)
2.  MAKE (the Accumulator increase its Present_Value)
```

```
3.  READ $V_2$ = Present_Value)
4.  READ $V_3$ = Present_Value)
5.  IF (the Accumulator is stopped) THEN
CHECK ($V_3$ = $V_2$ = Prescale-multiplier * pulse-count of signals generated by the measuring instrument) / Prescale-moduloDivide + $V_1$)
    ELSE
        CHECK ($V_1 < V_2 < V_3$)
```

7.3.2.X37.1.3 Logging_Record in Accumulator Test

Purpose: To verify the correct values represented in Logging_Record of Accumulator.

Test Concept: Two readings of the Logging_Object acquiring the Logging_Record are performed, PV_{prior} being the value from the first, and Present_Value matching what is observed in the second Logging_Record. Then all fields are checked to ensure these match the values expected.

Configuration Requirements: The IUT shall be configured so that Logging_Record capture is testable. If there is no Logging_Record property present in any Accumulator object, then this test shall be skipped.

Test Steps:

```
1.  MAKE (the Logging_Object acquire the Logging_Record)
2.  $PV_{prior}$ = present-value parameter in the Logging_Record
3.  MAKE (the Logging_Object acquire another Logging_Record)
4.  CHECK (Logging_Record list of values are:
        timestamp: the local date and time,
        present-value: Present_Value,
        accumulated-value: Present_Value - $PV_{prior}$,
        accumulated-status: NORMAL)
```

7.3.2.X37.1.4 Logging_Record in Accumulator RECOVERED Test

Purpose: To verify the correct values represented in Logging_Record of Accumulator after one or more writes to Value_Before_Change or Value_Set.

Test Concept: The effect of the Logging_Object acquiring the Logging_Record is checked to ensure that after one or more writes to Value_Before_Change or Value_Set, it matches the values expected.

Configuration Requirements: The IUT shall be configured so that Logging_Record capture is testable. If there is no Logging_Record property present in any Accumulator object, or if neither Value_Before_Change nor Value_Set is writable in an object which does have a Logging_Record property, then this test shall be skipped.

Test Steps:

```
1.  MAKE (the Logging_Object acquire the Logging_Record)
2.  $PV_{prior}$ = present-value parameter in the Logging_Record
3.  WRITE (either Value_Before_Change or Value_Set in the object that contains Logging_Record)
4.  MAKE (the Logging_Object acquire another Logging_Record)
5.  CHECK (Logging_Record list of values are:
        timestamp: the local date and time,
        present-value: Present_Value,
        accumulated-value: (Present_Value - Value_Set) +
                           (Value_Before_Change - $PV_{prior}$),

        accumulated-status: RECOVERED)
```

7.3.2.X37.1.5 Logging_Record in Accumulator STARTING Test

Purpose: To verify the correct values represented in Logging_Record of Accumulator when no data has been acquired since startup by the object identified by Logging_Object.

Test Concept: The Logging_Record is observed when no data has been acquired by the object identified by Logging_Object, to ensure that it matches the values expected.

Configuration Requirements: The IUT shall be in a state when no data has been acquired since startup by the object identified by Logging_Object. If there is no Logging_Record property present in any Accumulator object, then this test shall be skipped.

Test Steps:

```
1.  CHECK (Logging_Record list of values are:
        timestamp: unspecified,
        present-value: Present_Value,
        accumulated-value: 0,
        accumulated-status: STARTING)
3.  MAKE (the Logging_Object acquire the Logging_Record)
4.  CHECK (Logging_Record list of values are:
        timestamp: the local date and time,
        present-value: Present_Value,
        accumulated-value: same as present-value,
        accumulated-status: STARTING)
```

7.3.2.X37.1.6 Out_Of_Service Accumulator Test

Purpose: This test case verifies that Present_Value, Pulse_Rate, and the Reliability property are writable when Out_Of_Service is TRUE.

Test Concept: Select one instance of each appropriate object type and test it as described. Verify the interrelationship between the Out_Of_Service, Status_Flags, and Reliability properties. If the Out_Of_Service property of the object under test is not writable, and the value of the property cannot be changed by other means, then this test shall be omitted. If the Reliability property is not supported then step 5 shall be omitted.

Test Steps:

```
1.  IF (Out_Of_Service is writable) THEN
        WRITE Out_Of_Service = TRUE
    ELSE
        MAKE (Out_Of_Service TRUE)
2.  VERIFY Out_Of_Service = TRUE
3.  VERIFY Status_Flags = (?, FALSE, ?, TRUE)
4.  REPEAT X = (all values meeting the functional range requirements of 7.2.1) DO {
        WRITE Present_Value = X
        VERIFY Present_Value = X
        }
5.  IF (Reliability is present and writable) THEN
        REPEAT X = (all values of the Reliability enumeration appropriate to the object type except
                    NO_FAULT_DETECTED) DO {
            WRITE Reliability = X
            VERIFY Reliability = X
            VERIFY Status_Flags = (TRUE, TRUE, ?, TRUE)
            WRITE Reliability = NO_FAULT_DETECTED
            VERIFY Reliability = NO_FAULT_DETECTED
            VERIFY Status_Flags = (?, FALSE, ?, TRUE)
            }
6.  REPEAT X = (all values meeting the functional range requirements of 7.2.1) DO {
        WRITE Pulse_Rate = X
        VERIFY Pulse_Rate = X
        }
7.  IF (Out_Of_Service is writable) THEN
```

```
        WRITE Out_Of_Service = FALSE
    ELSE
        MAKE (Out_Of_Service FALSE)
8.  VERIFY Out_Of_Service = FALSE
9.  VERIFY Status_Flags = (?, ?, ?, FALSE)
```

7.3.2.X37.1.7 Value_Set Writing Test

Purpose: Verifying that writes to the Value_Set are reflected atomically into the object's properties.

Test Concept: Writing the Value_Set shall be reflected atomically in the Value_Set and Present_Value properties, while the old Present_Value is stored into the Value_Before_Change property, and the Value_Change_Time shall update.

Test Steps:

```
1.  READ OldV =  Present_Value
2.  WRITE Value_Set = (NewV, any valid value)
3.  VERIFY Value_Set = NewV
4.  VERIFY Present_Value = NewV
5.  VERIFY Value_Before_Change = OldV
6.  VERIFY Value_Change_Time = (approximately the current local time)
```

7.3.2.X37.1.8 Value_Before_Change Writing Test

Purpose: To verify the correct atomic operations of writing the Accumulator Value_Before_Change.

Test Concept: Write the Value_Before_Change and verify that it is reflected atomically in the Value_Before_Change property, while the old Present_Value is stored into the Value_Set property, and the Value_Change_Time shall update.

Test Steps:

```
1.  READ OldV =  Present_Value
2.  WRITE Value_Before_Change = (NewV, any valid value)
3.  VERIFY Value_Before_Change = NewV
4.  VERIFY Value_Set = OldV
5.  VERIFY Value_Change_Time = (approximately the current local time)
```

[In BTL Specified Tests, add new tests 8.4.X7 and 8.5.X7]

8.4.X7 UNSIGNED_RANGE ConfirmedEventNotification Test

Purpose: To verify the correct operation of the UNSIGNED_RANGE event algorithm. This test applies to Event Enrollment objects with an Event_Type of UNSIGNED_RANGE and to object types that generate this event type intrinsically.

Test Concept: This test is the same as 8.4.6 except that the Event_Type is UNSIGNED_RANGE instead of OUT_OF_RANGE, and there is no Deadband. If the pMonitoredValue property is not under the tester's control in IUT, then pHighLimit and/or pLowLimit are modified to generate Event notifications. The object begins the test in a NORMAL state. The pMonitoredValue is raised to a value that is above the high limit. After the time delay expires the object should enter the HIGH_LIMIT state and transmit an event notification message. The pMonitoredValue is lowered to a value that is below the high limit. After the time delay expires the object should enter the NORMAL state and issue an event notification. The same process is repeated to test the low limit.

Configuration Requirements: The IUT shall be configured such that the Event_Enable property has a value of TRUE for the TO_OFFNORMAL and TO_NORMAL transitions, if possible. pLimitEnable property shall have a value of TRUE for both HighLimit and LowLimit events, if possible. The 'Issue Confirmed Notifications' parameter in the Recipient_List of the configured Notification Class shall have a value of TRUE. The Recipient_List of the configured Notification Class shall contain recipients, thus ensuring that notifications are emitted. The event-generating objects shall be in a NORMAL state at the start of the test.

Test Steps:

```
1.  VERIFY pCurrentState = NORMAL
2.  IF (pMonitoredValue is writable) THEN
        WRITE pMonitoredValue = (a value x: (x > pHighLimit))
    ELSE
        MAKE (pMonitoredValue have a value x: (x > pHighLimit))
3.  WAIT (pTimeDelay)
4.  BEFORE Notification Fail Time
        RECEIVE ConfirmedEventNotification-Request,
            'Process Identifier' =            (any valid process ID),
            'Initiating Device Identifier' =  IUT,
            'Event Object Identifier' =       (the object being tested),
            'Time Stamp' =                    (Toffnormal: the current local time),
            'Notification Class' =            (the configured notification class),
            'Priority' =                      (the value configured to correspond to a TO_OFFNORMAL transition),
            'Event Type' =                    UNSIGNED_RANGE,
            'Notify Type' =                   EVENT | ALARM,
            'AckRequired' =                   TRUE | FALSE,
            'From State' =                    NORMAL,
            'To State' =                      HIGH_LIMIT,
            'Event Values' =                  pMonitoredValue, pStatusFlags, pHighLimit
5.  TRANSMIT BACnet-SimpleACK-PDU
6.  IF (the object being tested is not an Event Enrollment object OR
     (Protocol_Revision is present AND Protocol_Revision ≥ 13)) THEN
        VERIFY pStatusFlags = (TRUE, FALSE, ?, ?)
7.  VERIFY pCurrentState = HIGH_LIMIT
8.  IF (Protocol_Revision is present AND Protocol_Revision ≥ 1) THEN
        VERIFY Event_Time_Stamps = (Toffnormal, *, *)
9.  IF (pMonitoredValue is writable) THEN
        WRITE pMonitoredValue = (a value x: (pLowLimit  < x < pHighLimit))
    ELSE
        MAKE (pMonitoredValue have a value x: (pLowLimit < x < pHighLimit))
10. WAIT (pTimeDelayNormal)
11. BEFORE Notification Fail Time
        RECEIVE ConfirmedEventNotification-Request,
            'Process Identifier' =            (any valid process ID),
            'Initiating Device Identifier' =  IUT,
            'Event Object Identifier' =       (the object being tested),
            'Time Stamp' =                    (Tnormal: the current local time),
            'Notification Class' =            (the configured notification class),
            'Priority' =                      (the value configured to correspond to a TO_NORMAL transition),
            'Event Type' =                    UNSIGNED_RANGE,
            'Notify Type' =                   EVENT | ALARM,
            'AckRequired' =                   TRUE | FALSE,
            'From State' =                    HIGH_LIMIT,
            'To State' =                      NORMAL,
            'Event Values' =                  pMonitoredValue, pStatusFlags, pHighLimit
12. TRANSMIT BACnet-SimpleACK-PDU
13. IF (the object being tested is not an Event Enrollment object OR
     (Protocol_Revision is present AND Protocol_Revision ≥ 13)) THEN
        VERIFY pStatusFlags = (FALSE, FALSE, ?, ?)
14. VERIFY pCurrentState = NORMAL
15. IF (Protocol_Revision is present AND Protocol_Revision ≥ 1) THEN
        VERIFY Event_Time_Stamps = (Toffnormal, *, Tnormal)
16. IF (pMonitoredValue is writable) THEN
        WRITE pMonitoredValue = (a value x: (x < pLowLimit))
    ELSE
        MAKE (pMonitoredValue have a value x: (x < pLowLimit))
17. WAIT (pTimeDelay)
18. BEFORE Notification Fail Time
```

```
          RECEIVE ConfirmedEventNotification-Request,
               'Process Identifier' =             (any valid process ID),
               'Initiating Device Identifier' =   IUT,
               'Event Object Identifier' =        (the object being tested),
               'Time Stamp' =                     (Tlowlimit: the current local time),
               'Notification Class' =             (the configured notification class),
               'Priority' =                       (the value configured to correspond to a TO_OFFNORMAL transition),
               'Event Type' =                     UNSIGNED_RANGE,
               'Notify Type' =                    EVENT | ALARM,
               'AckRequired' =                    TRUE | FALSE,
               'From State' =                     NORMAL,
               'To State' =                       LOW_LIMIT,
               'Event Values' =                   pMonitoredValue, pStatusFlags, pLowLimit
19.  TRANSMIT BACnet-SimpleACK-PDU
20.  IF (the object being tested is not an Event Enrollment object OR
      (Protocol_Revision is present AND Protocol_Revision ≥ 13)) THEN
          VERIFY pStatusFlags = (TRUE, FALSE, ?, ?)
21.  VERIFY pCurrentState = LOW_LIMIT
22.  IF (Protocol_Revision is present AND Protocol_Revision ≥ 1) THEN
          VERIFY Event_Time_Stamps = (Tlowlimit, *, Tnormal)
23.  IF (pMonitoredValue is writable) THEN
          WRITE pMonitoredValue = (a value x: (Low_Limit < x < High_Limit))
     ELSE
          MAKE (pMonitoredValue have a value x: (Low_Limit < x < High_Limit))
24.  WAIT (pTimeDelayNormal)
25.  BEFORE Notification Fail Time
          RECEIVE ConfirmedEventNotification-Request,
               'Process Identifier' =             (any valid process ID),
               'Initiating Device Identifier' =   IUT,
               'Event Object Identifier' =        (the object being tested),
               'Time Stamp' =                     (Tlowtonormal: the current local time),
               'Notification Class' =             (the configured notification class),
               'Priority' =                       (the value configured to correspond to a TO_NORMAL transition),
               'Event Type' =                     UNSIGNED_RANGE,
               'Notify Type' =                    EVENT | ALARM,
               'AckRequired' =                    TRUE | FALSE,
               'From State' =                     LOW_LIMIT,
               'To State' =                       NORMAL,
               'Event Values' =                   pMonitoredValue, pStatusFlags, pLowLimit
26.  TRANSMIT BACnet-SimpleACK-PDU
27.  IF (the object being tested is not an Event Enrollment object OR
      (Protocol_Revision is present AND Protocol_Revision ≥ 13)) THEN
          VERIFY pStatusFlags = (FALSE, FALSE, ?, ?)
28.  VERIFY pCurrentState = NORMAL
29.  IF (Protocol_Revision is present AND Protocol_Revision ≥ 1) THEN
          VERIFY Event_Time_Stamps = (Tlowlimit, *, Tlowtonormal)
```

Notes to Tester: The 'Message Text' parameter is omitted in the test description because it is optional. The IUT may include this parameter in the notification messages. The time stamps indicated by "*" can have a value that indicates an unspecified time or a time that precedes the timestamp of the first received notification.

8.5.X7 UNSIGNED_RANGE UnconfirmedEventNotification Test

Purpose: To verify the correct operation of the UNSIGNED_RANGE event algorithm. This test applies to Event Enrollment objects with an Event_Type of UNSIGNED_RANGE and to object types that generate this event type intrinsically.

Configuration Requirements: The IUT shall be configured such that the Event_Enable property has a value of TRUE for the TO-OFFNORMAL and TO-NORMAL transitions, if possible. pLimitEnable shall have a value of TRUE for both HighLimit and LowLimit events, if possible. 'Issue Confirmed Notifications' parameter in the Recipient_List of the configured Notification Class shall have a value of FALSE. The Recipient_List of the configured Notification Class shall contain recipients, thus ensuring that notifications are emitted. The event-generating objects shall be in a NORMAL state at the start of the test.

Test Steps: The test steps for this test case are identical to the test steps in 8.4.X7 except that the ConfirmedEventNotification requests are UnconfirmedEventNotification requests and the TD does not acknowledge receiving the notifications.

Notes to Tester: The passing results for this test case are identical to the ones in 8.4.X7 except that the event notifications shall be conveyed using an UnconfirmedEventNotification service request. The MAC address used for these messages shall be either a broadcast that reaches the local network of the TD or the MAC address of the TD.