[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 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-TP14.0g-1: Directed Unicast Test Changes

Overview:

The Clarification Request **BTL-CRR-0149_10.X.2_non-BROADCAST.doc** was approved in September 2010. The question for this CR was whether or not to allow a Directed Unicast Who-Is in Step2.

Looking at the current version of this test, the test number has been changed to 10.7.2 Router Binding via Application Layer Services in 135.1-2013.

Most of the fix seem to have been incorporated already, but not all. In order to close this wID and move on, we need to review and see if all required changes are in correctly. Everything highlighted in yellow are the difference between what was on the CR and what was on the current 135.1-2013.

Changes:

[In BTL Specified Tests, Derive modified test 10.7.2 Router Binding via Application Layer Services from 135.1-2013]

10.7.2 Router Binding via Application Layer Services

Dependencies: ReadProperty Service Initiation Tests, 8.18, ReadProperty Service Execution Tests, 9.18, Who-Is Service Initiation Tests, 8.34

BACnet Reference Clause: 6.5.3

Purpose: To verify that the IUT can initiate requests to a remote network and respond to requests from a remote network after the IUT uses the Who-Is and I-Am Application Layer services to discover the MAC address of the router to that remote network.

Test Concept: The IUT broadcasts a Who-Is request to discover device D2A and notes the MAC address of the intervening router in the corresponding I-Am reply. The TD transmits a request to a device on the remote network and responds to a request from the remote network without performing any further form of dynamic router binding. If the IUT does not support application layer router binding, then this test shall be omitted. If the IUT cannot initiate a ReadProperty request, then another confirmed service can be substituted. The IUT may use the deviceInstanceRange form of Who-Is.

Clause 6.5.3 specifically mentions router binding via Who-Is and does not mention router binding by initiating other application layer services (such as Who-Has) or by lurking and noting the router MAC addresses for incoming application layer requests. For this reason the test only allows for router binding via Who-Is.

Test Steps:

```
1.  MAKE (IUT transmit Who-Is to discover the device on the remote network)
2.  RECEIVE
        DA = BROADCAST,
        SA = IUT,
        DNET =  GLOBAL BROADCAST,
        Hop Count = 255,
        BACnet-Unconfirmed-Request-PDU,
        'Service Choice' = who-Is
    | (DA = BROADCAST,
        SA = IUT,
        DNET = DNET2,
        DADR= BROADCAST, or D2A
        Hop Count = 255,
        BACnet-Unconfirmed-Request-PDU,
        'Service Choice' = who-Is  )
3.  TRANSMIT
        DA = BROADCAST,
        SA = TD,
        SNET = DNET2,
        SADR = D2A,
```

BACnet-Unconfirmed-Request-PDU,
‘Service Choice’ = I-Am,
'I Am Device Identifier' = (device object, instance number of D2A),
'Max APDU Length Accepted ' = (any valid value),
‘Segmentation Supported’ = (any valid value),
'Vendor ID ' = (any valid value)

4. MAKE (IUT transmit a ReadProperty request to the D2A device on the remote network)
5. RECEIVE
DA = TD,
SA = IUT,
DNET = DNET2,
DADR= D2A,
Hop Count = 255,
BACnet-Confirmed-Request-PDU,
'Service Choice' = ReadProperty-Request,
'Object Identifier' = (O1, any BACnet standard object in D2A),
'Property Identifier' = (P1, any required property of the specified object)
6. TRANSMIT
DA = IUT,
SA = TD,
SNET = DNET2,
SADR = D2A,
BACnet-ComplexACK-PDU,
‘Service ACK Choice’ = ReadProperty-ACK,
'Object Identifier' = O1,
'Property Identifier' = P1,
'Property Value' = (any valid value)

BTL-TP14.0g-2: Subscription Test Requirement Changes

Overview:

The response to CR-0299 called for our tests to be modified to allow a longer 'Lifetime'. The smallest amount "supported by the IUT", also makes more sense than a constant as the specified 'Lifetime' in several tests in 9.2.2.etc. in section 4.9.1 COV-A Base Requirements of Test Plan.

These same several 9.2.2.etc tests have overspecified in steps how to stop IUT from resubscribing. A different approach to MAKE IUT stop resubscribing, would be correct, as long as the result is "The IUT had at some earlier time expected ConfirmedCOVNotification with a particular Process Identifier and object identifier, but now doesn't expect to receive that.“

Wait more than 2* Lifetime if we are allowing very small Lifetime, because 2* epsilon is too small.

Also make a change for correctness, where in test 9.2.2.1, the TD needs to TRANSMIT ConfirmedCOVNotification-Request immediately after successful completion of a subscription or re-subscription request.

Also make a change for correctness, in all three tests, to refers to the step where the TD does TRANSMIT SubscribeCOV-Request with the correct step number, and add the suffix -Request.

These tests shall be Skipped if not configurable, since in COV-A the initiation of SubscribeCOV-Request with 'Issue Confirmed Notifications' equal to TRUE is entirely optional. BTL-WG in the existing versions with the steps specified only make sense in isolation, not in implementations of COV-A such as its use by T-VMT-E-B.

Changes:

[In BTL Specified Tests, derive modified tests 9.2.2.1, 9.2.2.2, and 9.2.2.4 from the versions in 135.1-2013]

9.2.2.1 Change of Value Notification Arrives after Subscription has Expired

Purpose: To verify that an appropriate error is returned if a COV notification arrives after the subscription time period has expired.

Test Steps:

Configuration Requirements: If the IUT does not support initiation of SubscribeCOV-Request with 'Issue Confirmed Notifications' equal to TRUE, then this test shall be skipped.

```
1.  RECEIVE SubscribeCOV-*Request*,
        'Subscriber Process Identifier' =        (any valid process identifier),
        'Monitored Object Identifier' =          (any object X of a type that supports COV notification),
        'Issue Confirmed Notifications ' =       TRUE,
        'Lifetime' =                             (~~a value no greater than one minute~~*any valid Lifetime*)
2.  TRANSMIT BACnet-SimpleACK-PDU
3.  *TRANSMIT ConfirmedCOVNotification-Request,*
            *'Subscriber Process Identifier' =   (the process identifier used in step 1),*
            *'Initiating Device Identifier' =    TD,*
            *'Monitored Object Identifier' =     X,*
            *'Time Remaining' =                  (any amount of time greater than 0),*
            *'List of Values' =                  (a list of values appropriate to object X)*
*4.  MAKE (the IUT stop resubscribing, if it resubscribes automatically)*
*5*~~3~~.  WAIT (~~a value two times~~ *at least* Lifetime*, but sufficient to ensure the subscription has expired*)
*6*~~4~~.  TRANSMIT ConfirmedCOVNotification-Request,
            'Subscriber Process Identifier' =    (the process identifier used in step ~~2~~*1*),
            'Initiating Device Identifier' =     TD,
            'Monitored Object Identifier' =      X,
            'Time Remaining' =                   (any amount of time greater than 0),
            'List of Values' =                   (a list of values appropriate to object X)
*7*~~5~~.  IF (Protocol_Revision is present and Protocol_Revision >= 10) THEN
        RECEIVE BACnet-Error-PDU,
                Error Class  =                   SERVICES,
                    Error Code  =                UNKNOWN_SUBSCRIPTION |
            (BACnet-SimpleACK-PDU)
ELSE
        RECEIVE BACnet-Error-PDU,
                    Error Class  =               SERVICES,
                    Error Code  =                (any valid error code for class SERVICES) |
            (BACnet-SimpleACK-PDU)
```

9.2.2.2 Change of Value Notifications with Invalid Process Identifier
Reason for Change: 135-2008h allows for a SimpleAck or a specific error code to return if a subscription does not exist.

Purpose: To verify that an appropriate error is returned if a COV notification arrives that contains a process identifier that does not match any current subscriptions.

Configuration Requirements: If the IUT does not support initiation of SubscribeCOV-Request with 'Issue Confirmed Notifications' equal to TRUE, then this test shall be skipped.

Test Steps:

```
1.  RECEIVE SubscribeCOV-Request,
        'Subscriber Process Identifier' =           (any valid process identifier),
        'Monitored Object Identifier' =             (any object X of a type that supports COV notification),
        'Issue Confirmed Notifications ' =          TRUE,
        'Lifetime' =                                (~~a value no greater than one minute~~any valid Lifetime)
2.  TRANSMIT BACnet-SimpleACK-PDU
3.  TRANSMIT ConfirmedCOVNotification-Request,
            'Subscriber Process Identifier' =    (a process identifier different from the one used in step ~~2~~1),
            'Initiating Device Identifier' =     TD,
            'Monitored Object Identifier' =      X,
            'Time Remaining' =                   (any amount of time greater than 0),
            'List of Values' =                   (a list of values appropriate to object X)
4.  IF (Protocol_Revision is present and Protocol_Revision >= 10) THEN
        RECEIVE BACnet-Error-PDU,
                    Error Class  =               SERVICES,
                    Error Code  =                UNKNOWN_SUBSCRIPTION |
            (BACnet-SimpleACK-PDU)
ELSE
        RECEIVE BACnet-Error-PDU,
                    Error Class  =               SERVICES,
                    Error Code  =                (any valid error code for class SERVICES) |
            (BACnet-SimpleACK-PDU)
```

9.2.2.4 Change of Value Notifications with Invalid Monitored Object Identifier
Reason for Change: 135-2008h allows for a SimpleAck or a specific error code to return if a subscription does not exist.

Purpose: To verify that an appropriate error is returned if a COV notification arrives that contains a monitored object identifier that does not match any current subscriptions.

Configuration Requirements: If the IUT does not support initiation of SubscribeCOV-Request with 'Issue Confirmed Notifications' equal to TRUE, then this test shall be skipped.

Test Steps:

```
1.  RECEIVE SubscribeCOV-Request,
        'Subscriber Process Identifier' =           (any valid process identifier),
        'Monitored Object Identifier' =             (any object X of a type that supports COV notification),
        'Issue Confirmed Notifications ' =          TRUE,
        'Lifetime' =                                (~~a value no greater than one minute~~any valid Lifetime)
2.  TRANSMIT BACnet-SimpleACK-PDU
3.  TRANSMIT ConfirmedCOVNotification-Request,
```

'Subscriber Process Identifier' = (the process identifier used in step ~~2~~*1*),
'Initiating Device Identifier' = TD,
'Monitored Object Identifier' = (any object Y *in the IUT* ~~supporting COV notification~~ except X*, and for which IUT does not already have an active subscription*),
'Time Remaining' = (any amount of time greater than 0),
'List of Values' = (a list of values appropriate to object Y)

```
4.    IF (Protocol_Revision is present and Protocol_Revision >= 10) THEN
          RECEIVE BACnet-Error-PDU,
                    Error Class  =          SERVICES,
                    Error Code  =           UNKNOWN_SUBSCRIPTION |

                    (BACnet-SimpleACK-PDU)
ELSE
          RECEIVE BACnet-Error-PDU,
                    Error Class  =          SERVICES,
                    Error Code  =           (any valid error code for class SERVICES) |
                    (BACnet-SimpleACK-PDU)
```

Notes to Tester: If possible, select an object Y for which IUT supports COV Subscription.

[In BTL Test Plan, add Test Conditionality to existing tests in sections 4.9.1 DS-COV-A Base Requirements, and revise the test references, as indicated.]

4.9 Data Sharing - COV -A

4.9.1 Base Requirements

Base requirements must be met by any IUT claiming conformance to this BIBB.

~~135.1-2013~~BTL - 9.2.2.1 - Change of Value Notification Arrives after Subscription has Expired		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests~~ASHRAE 135.1-2013~~***.
	Test Conditionality	Must be executed if IUT supports initiation of SubscribeCOV-Request with 'Issue Confirmed Notifications' equal to TRUE.
	Test Directives	
	Testing Hints	
	Notes & Results	

~~135.1-2013~~BTL - 9.2.2.2 - Change of Value Notifications with Invalid Process Identifier		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests~~ASHRAE 135.1-2013~~***.
	Test Conditionality	Must be executed if IUT supports initiation of SubscribeCOV-Request with 'Issue Confirmed Notifications' equal to TRUE.
	Test Directives	
	Testing Hints	
	Notes & Results	

~~135.1-2013~~BTL - 9.2.2.4 - Change of Value Notifications with Invalid Monitored Object Identifier		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests~~ASHRAE 135.1-2013~~***.
	Test Conditionality	Must be executed if IUT supports initiation of SubscribeCOV-Request with 'Issue Confirmed Notifications' equal to TRUE.
	Test Directives	
	Testing Hints	
	Notes & Results	

BTL-TP14.0g-3: Clarifies CHANGE_OF_CHARACTERSTRING Tests

Overview:

Clarification Request BTL-CR-0322 pointed out a problem with test 8.4.X4 CHANGE_OF_CHARACTERSTRING Tests (ConfirmedEventNotification).

Changes:

[In BTL Specified Tests, Modify test 8.4.X4]

8.4.X4 CHANGE_OF_CHARACTERSTRING Tests (ConfirmedEventNotification)

Reason for Change: New algorithm for Protocol_Revision 10.

Dependencies: ReadProperty Service Execution Tests, 9.18; WriteProperty Service Execution Tests, 9.22.

BACnet Reference Clauses: <update these as appropriate>

Purpose: To verify the correct operation of the CHANGE_OF_CHARACTERSTRING event algorithm. This test applies to Event Enrollment objects with an Event_Type of CHANGE_OF_CHARACTERSTRING and to intrinsic event reporting for CharacterString Value objects.

Test Concept: The object begins the test in a NORMAL state. The Present_Value (referenced property) is changed to a value that is one of the values designated in List_Of_Values. After the time delay expires the object should enter the OFFNORMAL state and transmit an event notification message. The Present_Value (referenced property) is then changed to a different value in the List_Of_Values. After the time delay expires the object should enter the OFFNORMAL state and transmit an event notification message. The Present_Value (referenced property) is then changed to a value corresponding to a NORMAL state. After the time delay the object should enter the NORMAL state and transmit an event notification message. The transition to and from FAULT is also tested.

Configuration Requirements: The IUT shall be configured such that the Event_Enable property has a value of TRUE for the TO-OFFNORMAL, TO-FAULT and TO-NORMAL transitions. The Issue_Confirmed_Notifications property shall have a value of TRUE. The event-generating objects shall be in a NORMAL state at the start of the test.

The object shall be configured with a non-empty Alarm_Values property and a non-empty Fault_Values property if possible.

In the test description below Present_Value is used as the referenced property. If an Event Enrollment object is being tested Present_Value should be replaced by the appropriate property reference.

Test Steps:

```
1.  VERIFY Event_State = NORMAL
2.  IF (the object, or referenced object, if using Event Enrollment, has a non-empty Alarm_Values property) THEN
3.      IF (Present_Value is writable) THEN
            WRITE Present_Value = (a value x: x = one of the Alarm_Values)
        ELSE
            MAKE (Present_Value have a value x: x = one of the Alarm_Values)
4.      WAIT (Time_Delay)
5.      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 or the Event Enrollment 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' =                    CHANGE_OF_CHARACTERSTRING,
            'Notify Type' =                   EVENT | ALARM,
            'AckRequired' =                TRUE | FALSE,
            'From State' =                 NORMAL,
            'To State' =                   OFFNORMAL,
            'Event Values' =               Present_Value, Status_Flags
6.      TRANSMIT BACnet-SimpleACK-PDU
7.      IF (the object being tested is NOT an Event Enrollment object) THEN
                VERIFY Status_Flags = (TRUE, FALSE,?,?)
8.      VERIFY Event_State = OFFNORMAL
9.      VERIFY Event_Time_Stamps = (Toffnormal, *, *)

10. IF (the object, or referenced object, if using Event Enrollment, has a Alarm_Values property with more than 1 entry)
THEN
11.     IF (Present_Value is writable) THEN
            WRITE Present_Value = (a value x: x = one of the Alarm_Values not used in prior steps)
        ELSE
            MAKE (Present_Value have a value x: x = one of the Alarm_Values not used in prior steps)
12.     WAIT (Time_Delay)
13.     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 or the Event Enrollment 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' =                 CHANGE_OF_CHARACTERSTRING,
            'Notify Type' =                EVENT | ALARM,
            'AckRequired' =                TRUE | FALSE,
            'From State' =                 OFFNORMAL,
            'To State' =                   OFFNORMAL,
            'Event Values' =               Present_Value, Status_Flags
14.     TRANSMIT BACnet-SimpleACK-PDU
15.     IF (the object being tested is NOT an Event Enrollment object) THEN
                VERIFY Status_Flags = (TRUE, FALSE,?,?)
16.     VERIFY Event_State = OFFNORMAL
17.     VERIFY Event_Time_Stamps = (Toffnormal, *, *)
18. IF (the object, or referenced object, if using Event Enrollment, has a non-empty Alarm_Values property) THEN
19.     IF (Present_Value is writable) THEN
            WRITE Present_Value = (a value x: x corresponds to a NORMAL state)
        ELSE
            MAKE (Present_Value have a value x: x corresponds to a NORMAL state)
20.     WAIT (Time_Delay)
21.     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 or the object referenced by the Event
                                           Enrollment 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' =                 CHANGE_OF_CHARACTERSTRING,
            'Notify Type' =                EVENT | ALARM,
            'AckRequired' =                TRUE | FALSE,
            'From State' =                 OFFNORMAL,
            'To State' =                   NORMAL,
            'Event Values' =               Present_Value, Status_Flags
22.     TRANSMIT BACnet-SimpleACK-PDU
23.     IF (the object being tested is NOT an Event Enrollment object) THEN
```

```
                VERIFY Status_Flags = (FALSE, FALSE, ?, ?)
24.     VERIFY Event_State = NORMAL
25.     VERIFY Event_Time_Stamps = (Toffnormal, *, Tnormal)
26.  IF (the object, or referenced object, if testing Event Enrollment, is configured with a non-empty Fault_Values property)
THEN
27.     IF (Present_Value is writable) THEN
                WRITE Present_Value = (a value x: x = one of the Fault_Values)
        ELSE
                MAKE (Present_Value have a value x: x = one of the Fault_Values)
28.     WAIT (Time_Delay)
29.     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' =                    (Tfault: the current local time),
                'Notification Class' =            (the configured notification class),
                'Priority' =                      (the value configured to correspond to a TO-FAULT transition),
                'Event Type' =                    CHANGE_OF_CHARACTERSTRING,
                'Notify Type' =                   EVENT | ALARM,
                'AckRequired' =                   TRUE | FALSE,
                'From State' =                    NORMAL,
                'To State' =                      FAULT,
                'Event Values' =                  Present_Value, Status_Flags
30.             TRANSMIT BACnet-SimpleACK-PDU
31.     IF (the object being tested is NOT an Event Enrollment object) THEN
                VERIFY Status_Flags = (TRUE, TRUE, ?, ?)
32.     VERIFY Event_State = FAULT
33.     VERIFY Event_Time_Stamps = (Toffnormal, Tfault, Tnormal)
34.     VERIFY Reliability = MULTI_STATE_FAULT
35.     IF (Present_Value is writable) THEN
            WRITE Present_Value = (a value x: x corresponds to a NORMAL state)
        ELSE
            MAKE (Present_Value have a value x: x corresponds to a NORMAL state)
36.     WAIT (Time_Delay)
37.     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' =                    (Tfault: the current local time),
                'Notification Class' =            (the configured notification class),
                'Priority' =                      (the value configured to correspond to a TO-NORMAL transition),
                'Event Type' =                    CHANGE_OF_CHARACTERSTRING,
                'Notify Type' =                   EVENT | ALARM,
                'AckRequired' =                   TRUE | FALSE,
                'From State' =                    FAULT,
                'To State' =                      NORMAL,
                'Event Values' =                  Present_Value, Status_Flags
38.     TRANSMIT BACnet-SimpleACK-PDU
39.     IF (the object being tested is NOT an Event Enrollment object) THEN
                VERIFY Status_Flags = (FALSE, FALSE, ?, ?)
40.  VERIFY Event_State = NORMAL
41.  VERIFY Event_Time_Stamps = (Toffnormal, Tfault, Tnormal)
```

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.

BTL-TP14.0g-4: Clarifies Limit_Enable Tests

Overview:

The existing test is extremely long (58 steps) and tests more than the Limit_Enable property's ability to enable or disable the reporting of out-of-range events. In addition, there are some If/Else statements that can be replaced with a single MAKE statement. The proposed change breaks the existing test into two tests; one for lowLimitEnable, the other for highLimitEnable. This approach allows the conditionality of achievable Limit_Enable property values to be identified in the test plan and greatly reduces the number of steps in each test.

In addition, new Integer objects have been added to the standard that may have a writable Limit_Enable property and these objects are not currently covered.

Changes:

[In BTL Specified Tests, add these new tests]

7.3.1.13.X1 Limit_Enable Test, LowLimitEnable

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

Test Concept: The LowLimitEnable flag is set to true in the Limit_Enable property and the event-triggering property is manipulated to cause the low limit to be exceeded. This should generate an event notification and make Event_State = Low_Limit. After the event-triggering property is returned to a normal value, the LowLimitEnable flag is the set to false and the event-triggering property is again manipulated to exceed the low limit. No event notification should be observed and the Event_State must have a value of normal.

Configuration Requirements: Configure the object with pHighLimit, pLowLimit and pDeadband values such that pLowLimit + pDeadband < pHighLimit and both the pLowLimit and pHighLimit values are within the valid range of values for the event-triggering property. If the device cannot be configured with limit values that meet these conditions, then this test shall be skipped. The Event_Enable property shall be set to (TRUE, ?, TRUE) for this test. If the Event_Enable property cannot be configured such that the TO-NORMAL and the TO-OFFNORMAL transitions are TRUE, this test shall be skipped.

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

Test Steps:

```
1. MAKE pLimitEnable = (TRUE, ?)
2. VERIFY pCurrentState = NORMAL
3. MAKE (pMonitoredValue a value less than pLowLimit)
4. WAIT (pTimeDelay)
5.  BEFORE Notification Fail Time
        RECEIVE ConfirmedEventNotification-Request,
            'Process Identifier' =        (any valid process ID),
            'Initiating Device Identifier' = IUT,
            'Event Object Identifier' =   (the object configured for this test),
            'Time Stamp' =                (the current local time),
            'Notification Class' =        (the class corresponding to the object being tested),
            'Priority' =                  (the value configured to correspond to a
                                          TO-OFFNORMAL transition),
            'Event Type' =                OUT_OF_RANGE,
            'Notify Type' =               ALARM | EVENT,
            'AckRequired' =               TRUE | FALSE,
            'From State' =                NORMAL,
            'To State' =                  LOW_LIMIT,
            'Event Values' =              (values appropriate to the event type)
6. TRANSMIT SimpleAck-PDU
7. VERIFY pCurrentState = LOW_LIMIT
8. MAKE (pMonitoredValue a value that is between pLowLimit + pDeadband and pHighLimit)
9. WAIT (pTimeDelayNormal)
10. BEFORE Notification Fail Time
        RECEIVE ConfirmedEventNotification-Request,
            'Process Identifier' =          (any valid process ID),
            'Initiating Device Identifier' = IUT,
            'Event Object Identifier' = (the object configured for this test),
            'Time Stamp' =                  (the current local time),
            'Notification Class' =(the class corresponding to the object being tested),
            'Priority' =                    (the value configured to correspond to a
                                                  TO-NORMAL transition),
            'Event Type' =                  OUT_OF_RANGE,
            'Notify Type' =                 ALARM | EVENT,
            'AckRequired' =                 TRUE | FALSE,
            'From State' =                  LOW_LIMIT,
            'To State' =                    NORMAL,
            'Event Values' =                (values appropriate to the event type)
11. TRANSMIT SimpleAck-PDU
12. MAKE pLimitEnable = (FALSE, ?)
13. VERIFY pCurrentState = NORMAL
14. MAKE (pMonitoredValue a value less than pLowLimit)
15. WAIT (pTimeDelay + Notification Fail Time)
16. CHECK (verify that no notification message was transmitted)
17. VERIFY pCurrentState = NORMAL
```

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

7.3.1.13.X2 Limit_Enable Test, HighLimitEnable

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

Test Concept: The HighLimitEnable flag is set to true in the Limit_Enable property and the event-triggering property is manipulated to cause the high limit to be exceeded. This should generate an event notification and make Event_State = High_Limit. After the event-triggering property is returned to a normal value, the HighLimitEnable flag is the set to false and the event-triggering property is again manipulated to exceed the high limit. No event notification should be observed and the Event_State must have a value of normal.

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

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

Test Steps:

```
1. MAKE pLimitEnable = (?, TRUE)
2. VERIFY pCurrentState = NORMAL
3. MAKE (pMonitoredValue a value that exceeds pHighLimit)
4. WAIT (pTimeDelay)
5.   BEFORE Notification Fail Time
            RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' =          (any valid process ID),
                'Initiating Device Identifier' = IUT,
                'Event Object Identifier' = (the object configured for this test),
                'Time Stamp' =                  (the current local time),
                'Notification Class' =          (the class corresponding to the object being tested),
                'Priority' =                    (the value configured to correspond to a
                                                TO-OFFNORMAL transition),
                'Event Type' =                  OUT_OF_RANGE,
                'Notify Type' =                 ALARM | EVENT,
                'AckRequired' =                 TRUE | FALSE,
                'From State' =                  NORMAL,
                'To State' =                HIGH_LIMIT,
                'Event Values' =            (values appropriate to the event type)
6. TRANSMIT SimpleAck-PDU
7. VERIFY pCurrentState = HIGH_LIMIT
8. MAKE (pMonitoredValue a value that is between pLowLimit and pHighLimit - pDeadband)
9. WAIT (pTimeDelayNormal)
10. BEFORE Notification Fail Time
        RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' =      (any valid process ID),
                'Initiating Device Identifier' = IUT,
                'Event Object Identifier' = (the object configured for this test),
                'Time Stamp' =              (the current local time),
                'Notification Class' =(the class corresponding to the object being tested),
                'Priority' =                (the value configured to correspond to a
                                                TO-NORMAL transition),
                'Event Type' =              OUT_OF_RANGE,
                'Notify Type' =             ALARM | EVENT,
                'AckRequired' =             TRUE | FALSE,
                'From State' =              HIGH_LIMIT,
                'To State' =                NORMAL,
                'Event Values' =            (values appropriate to the event type)
11. TRANSMIT SimpleAck-PDU
12. MAKE pLimitEnable = (?, FALSE)
13. VERIFY pCurrentState = NORMAL
14. MAKE (pMonitoredValue a value that exceeds pHighLimit)
15. WAIT (pTimeDelay + Notification Fail Time)
16. CHECK (verify that no notification message was transmitted)
17. VERIFY pCurrentState = NORMAL
```

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

[In BTL Checklist, add an entry in AE-N-I-B]

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	Implements 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 form
	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
	O	Supports Event_Message_Texts property
	O	Supports Event_Message_Texts_Config property
	O	Implements intrinsic alarming in an Integer object
		[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.

[In BTL Test Plan, make the following changes in 5.2.19]

5.2.19 Implements Intrinsic Alarming in an Analog object

The IUT contains, or can be made to contain, an Analog object that can generate ConfirmedEventNotifications and UnconfirmedEventNotifications and thus contains a Limit_Enable property.

[Remove this entry]

~~**BTL - 7.3.1.13 - Limit_Enable Test**~~		
	~~**Test Method**~~	~~Manual~~
	~~**Configuration**~~	~~As per ***BTL Specified Tests***.~~
	~~**Test Conditionality**~~	~~Must be executed if Limit_Enable is writable.~~
	~~**Test Directives**~~	
	~~**Testing Hints**~~	~~If the Limit_Enable property is not writable, then the tester should apply those sections of this test that will apply to an object with the specified Limit_Enable value.~~
	~~**Notes & Results**~~	

[Add these entries]

BTL - 7.3.1.13.X1 Limit_Enable Test, LowLimitEnable		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If Limit_Enable is not configurable, this test may be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.1.13.X2 Limit_Enable Test, HighLimitEnable		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If Limit_Enable is not configurable, this test may be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	

[In BTL Test Plan, add the following]

5.2.27 Implements Intrinsic Alarming in an Integer object

The IUT contains, or can be made to contain, an Integer object that can generate ConfirmedEventNotifications and UnconfirmedEventNotifications and thus contains a Limit_Enable property.

BTL - 7.3.1.13.X1 Limit_Enable Test, LowLimitEnable

	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If Limit_Enable is not configurable, this test may be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	

BTL - 7.3.1.13.X2 Limit_Enable Test, HighLimitEnable

	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If Limit_Enable is not configurable, this test may be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	

BTL-TP14.0g-5: Clarifies How a Device can Fallback

Overview:
The existing test specifies exclusively accessing one or more properties of a single object in the TD. The scenario of accessing one or more properties of multiple objects in the TD also deserves testing coverage in our Test Package.

[In BTL Test Plan, revise the test reference in 4.3.9 section as indicated.]

4.3.9 Can Fallback to ReadProperty Based on Protocol_Services_Supported

The IUT is able to select between initiating ReadPropertyMultiple and ReadProperty requests based on whether or not support for ReadPropertyMultiple is advertised by a peer device's Protocol_Services_Supported property .

BTL~~135.1-2013~~ - 8.20.5.1 - The IUT Determines the TD does not Support the ReadPropertyMultiple Service		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests~~ASHRAE 135-2013~~***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

[In BTL Specified Tests, derive test 8.20.5.1 from the version in 135-2013]

8.20.5 Cases In Which ReadProperty Shall Be Used After ReadPropertyMultiple Fails

The tests defined in this clause are used to verify that an IUT which intiates ReadPropertyMultiple is able to obtain external property values via the ReadProperty service when interoperating with a device that does not support the ReadPropertyMultiple service.

8.20.5.1 The IUT Determines the TD does not Support the ReadPropertyMultiple Service

Purpose: Verifies the IUT's ability to automatically change its service choice from ReadPropertyMultiple to ReadProperty when the IUT determines the TD does not support the ReadPropertyMultiple service.

Test Concept: The IUT is configured in a manner that would normally cause it to access one or more properties in the TD via the ReadPropertyMultiple service. Prior to sending a ReadPropertyMultiple request, however, the IUT determines that the TD does not support the ReadPropertyMultiple service. The IUT instead attempts to access the TD's property values via the ReadProperty service (it is assumed that the IUT will make this determination by reading the TD's Protocol_Services_Supported property, but this test specifically does not attempt to verify this behavior).

Configuration Requirements: The TD is configured so that it does not support the ReadPropertyMultiple service. The IUT is configured such that it is ~~capable of~~ accessing one or more properties of a single *or multiple* object*s* in the TD via the ReadProperty and ReadPropertyMultiple services. ~~If the IUT cannot be configured in this way, then this test shall be omitted.~~

Test Steps:

```
1.  MAKE (a condition in the IUT that would normally cause it to send a ReadPropertyMultiple request to the TD to
            access one or more propert*y*~~ies~~ values~~ of a single object~~)
2.  WAIT (a time interval specified by the vendor as sufficient for the IUT to determine that the TD does not support the
            ReadPropertyMultiple service)
3.  REPEAT X = (the properties that the IUT is to read) DO {
        RECEIVE ReadProperty-Request,
            'Object Identifier' =           (object identifier referenced by X),
            'Property Identifier' =         (property identifier referenced by X)
        TRANSMIT ReadProperty-Ack,
            'Object Identifier' =           (object identifier referenced by X),
```

'Property Identifier' =	(property identifier referenced by X),
'Property Value' =	(any valid value)