[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-TP14.0b-1: Correct Timestamp expectation in Acked_Transition Tests, pg 2. [wID0917]
BTL-TP14.0b-2: Correct SubscribeCOV requirements, **pg 6**. [wID0839]
BTL-TP14.0b-3: Weaken Exception_Schedule length requirement, pg 8. [wID0874]
BTL-TP14.0b-4: Clarify WritePropertyMultiple Reject Test, pg 10. [wID0880]
BTL-TP14.0b-5: Correct Trigger Verification Test, pg 11. [wID0897]
BTL-TP14.0b-6: Correct Unsupported Message Text Character Set AcknowledgeAlarm Test, pg 13. [wID0916]
BTL-TP14.0b-7: Weaken Enumerated type Requirement, pg 15. [w0ID0888]

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 14.0b-1: Correct Timestamp expectation in Acked_Transition Test

Overview:

Clause 13.5.2 excerpt states: “The Time Stamp conveyed in the acknowledgment notification shall not be derived from the Time Stamp of the original event notification, but rather the time at which the acknowledgment notification is generated.” This is new language, not present in 135-2008, but which is presented as existing text in 135-2010*af*.

Some existing test specifications explicitly contradict this, and need corrections.

Changes:

[In BTL Specified Tests, derive from existing test 135.1-2013 - 7.3.1.11]

7.3.1.11 Acked_Transitions Tests

Dependencies: ConfirmedEventNotification Service Initiation Tests, 8.4; UnconfirmedEventNotification Service Initiation Tests, 8.5; AcknowledgeAlarm Service Execution Tests, 9.1; ReadProperty Service Execution Tests, 9.18; WriteProperty Service Execution Tests, 9.22.

BACnet Reference Clauses: 12.1.28, 12.2.24, 12.3.25, 12.4.21, 12.6.23, 12.7.27, 12.8.25, 12.12.11, 12.15.20, 12.16.20,12.17.35, 12.18.18, 12.19.19, 12.20.19, 12.23.27 and 12.25.23.

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. This test applies to Event Enrollment objects and Accumulator, Analog Input, Analog Output, Analog Value, Binary Input, Binary Output, Binary Value, Life Safety Point, Life Safety Zone, Loop, Multi-state Input, Multi-state Output, Multi-state Value, Pulse COnverter and Trend Log objects that support intrinsic reporting.

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).

In the test description below, “X” is used to designate the event-triggering property.

Test Steps:

1. WAIT (Time_Delay + **Notification Fail Time**)
2. VERIFY Event_State = NORMAL
3. VERIFY Acked_Transitions = (TRUE, TRUE, TRUE)
4. VERIFY Status_Flags = (FALSE, FALSE, ?, ?)
5. IF (X is writable) THEN
   WRITE X = (a value that is OFFNORMAL)
   ELSE
   MAKE (X have a value that is OFFNORMAL)
6. WAIT (Time_Delay)
7. BEFORE **Notification Fail Time**
   RECEIVE ConfirmedEventNotification-Request,
   'Process Identifier' = (any valid process ID),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (the 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-OFFNORMAL transition),
'Event Type' = (any valid event type),
'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)

8. TRANSMIT BACnet-SimpleACK-PDU
9. VERIFY Event_State = OFFNORMAL
10. VERIFY Acked_Transitions = (FALSE, TRUE, TRUE)
11. VERIFY Status_Flags = (TRUE, FALSE, ?, ?)
12. IF (X is writable) THEN
    WRITE X = (a value that is NORMAL)
    ELSE
    MAKE (X have a value that is NORMAL)
13. WAIT (Time_Delay)
14. BEFORE **Notification Fail Time**
    RECEIVE ConfirmedEventNotification-Request,
    'Process Identifier' = (any valid process ID),
    'Initiating Device Identifier' = IUT,
    'Event Object Identifier' = (the 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-NORMAL transition),
    'Event Type' = (any valid event type),
    '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)
15. TRANSMIT BACnet-SimpleACK-PDU
16. VERIFY Event_State = NORMAL
17. VERIFY Acked_Transitions = (FALSE, TRUE, FALSE)
18. VERIFY Status_Flags = (FALSE, FALSE, ?,?)
19. IF (the event-triggering object can be placed into a fault condition) THEN
20. MAKE (the event-triggering object change to a fault condition)
21. BEFORE **Notification Fail Time**
    RECEIVE ConfirmedEventNotification-Request,
    'Process Identifier' = (any valid process ID),
    'Initiating Device Identifier' = IUT,
    'Event Object Identifier' = (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),
    '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)
22. TRANSMIT BACnet-SimpleACK-PDU
23. VERIFY Event_State = FAULT
24. VERIFY Acked_Transitions = (FALSE, FALSE, FALSE)
25. VERIFY Status_Flags = (TRUE, TRUE, ?, ?)
26. MAKE (the event-triggering object change to a normal condition)
27. BEFORE **Notification Fail Time**
    RECEIVE ConfirmedEventNotification-Request,
    'Process Identifier' = (any valid process ID),

'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (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-NORMAL transition),
'Event Type' = (any valid event type),
'Notify Type' = (the notify type configured for this event),
'AckRequired' = TRUE,
'From State' = FAULT,
'To State' = NORMAL,
'Event Values' = (values appropriate to the event type)

28. TRANSMIT BACnet-SimpleACK-PDU
29. VERIFY Event_State = NORMAL
30. VERIFY Acked_Transitions = (FALSE, FALSE, FALSE)
31. VERIFY Status_Flags = (FALSE, FALSE, ?, ?)
32. TRANSMIT AcknowledgeAlarm-Request,
'Acknowledging Process Identifier' = (the value of the 'Process Identifier' in step 21),
'Event Object Identifier' = (the 'Event Object Identifier' in step 21),
'Event State Acknowledged' = FAULT,
'Time Stamp' = (the 'Time Stamp' in step 21),
'Time of Acknowledgment' = (the TD's current time)
33. RECEIVE BACnet-SimpleACK-PDU
34. IF (Protocol_Revision is present and Protocol_Revision ≥ 1) THEN
BEFORE **Notification Fail Time**
RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (the value of the 'Process Identifier' in step 21),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the 'Event Object Identifier' in step 21),
'Time Stamp' = (the *current time or sequence number*~~'Time Stamp' in step 21~~),
'Notification Class' = (the 'Notification Class' in step 21),
'Priority' = (the 'Priority' in step 21),
'Event Type' = (the 'Event Type' in step 21),
'Notify Type' = ACK_NOTIFICATION,
'To State' = FAULT
ELSE
BEFORE **Notification Fail Time**
. RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (the value of the 'Process Identifier' in step 21),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the 'Event Object Identifier' in step 21),
'Time Stamp' = (the *current time or sequence number*~~'Time Stamp' in step 21~~),
'Notification Class' = (the 'Notification Class' in step 21),
'Priority' = (the 'Priority' in step 21),
'Event Type' = (the 'Event Type' in step 21),
'Notify Type' = ACK_NOTIFICATION
35. TRANSMIT BACnet-SimpleACK-PDU
36. VERIFY Acked_Transitions = (FALSE, TRUE, FALSE)
37. TRANSMIT AcknowledgeAlarm-Request,
'Acknowledging Process Identifier' = (the value of the 'Process Identifier' in step 27),
'Event Object Identifier' = (the 'Event Object Identifier' in step 27),
'Event State Acknowledged' = NORMAL,
'Time Stamp' = (the 'Time Stamp' in step 27),
'Time of Acknowledgment' = (the TD's current time)
38. RECEIVE BACnet-SimpleACK-PDU
39. IF (Protocol_Revision is present and Protocol_Revision ≥ 1) THEN
BEFORE **Notification Fail Time**
RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (the value of the 'Process Identifier' in step 27),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the 'Event Object Identifier' in step 27),

'Time Stamp' = (the current time or sequence number),
'Notification Class' = (the 'Notification Class' in step 27),
'Priority' = (the 'Priority' in step 27),
'Event Type' = (the 'Event Type' in step 27),
'Notify Type' = ACK_NOTIFICATION,
'To State' = NORMAL

ELSE

BEFORE **Notification Fail Time**

RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (the value of the 'Process Identifier' in step 27),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the 'Event Object Identifier' in step 27),
'Time Stamp' = (the current time or sequence number),
'Notification Class' = (the 'Notification Class' in step 27),
'Priority' = (the 'Priority' in step 27),
'Event Type' = (the 'Event Type' in step 27),
'Notify Type' = ACK_NOTIFICATION

40. TRANSMIT BACnet-SimpleACK-PDU
41. VERIFY Acked_Transitions = (FALSE, TRUE, TRUE)
42. TRANSMIT AcknowledgeAlarm-Request,
'Acknowledging Process Identifier' = (the value of the 'Process Identifier' in step 7),
'Event Object Identifier' = (the 'Event Object Identifier' in step 7),
'Event State Acknowledged' = OFFNORMAL,
'Time Stamp' = (the 'Time Stamp' in step 7),
'Time of Acknowledgment' = (the TD's current time)
43. RECEIVE BACnet-SimpleACK-PDU
44. IF (Protocol_Revision is present and Protocol_Revision ≥ 1) THEN

BEFORE **Notification Fail Time**

RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (the value of the 'Process Identifier' in step 7),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the 'Event Object Identifier' in step 7),
'Time Stamp' = (the current time or sequence number),
'Notification Class' = (the 'Notification Class' in step 7),
'Priority' = (the 'Priority' in step 7),
'Event Type' = (the 'Event Type' in step 7),
'Notify Type' = ACK_NOTIFICATION,
'To State' = OFFNORMAL

ELSE

BEFORE **Notification Fail Time**

RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (the value of the 'Process Identifier' in step 7),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the *'Event Object Identifier' in step 7*~~current time or sequence number~~),
'Time Stamp' = (the *current time or sequence number*~~'Time Stamp' in step 7~~),
'Notification Class' = (the 'Notification Class' in step 7),
'Priority' = (the 'Priority' in step 7),
'Event Type' = (the 'Event Type' in step 7),
'Notify Type' = ACK_NOTIFICATION

45. TRANSMIT BACnet-SimpleACK-PDU
46. VERIFY Acked_Transitions = (TRUE, TRUE, TRUE)

Notes to Tester: The UnconfirmedEventNotification service may be substituted for the ConfirmedEventNotification service. 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 Test Plan, update referenced test from 135.1 to BTL]

BTL 14.0b-2: Correct SubscribeCOV requirements

Overview

Devices do not always have an object where executing a SubscribeCOV would return OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED.

Changes

[In BTL Specified Tests, modify existing test 9.10.2.1 derived from 135.1-2013 version, as indicated]

9.10.2.1 The Monitored Object Does Not Support COV Notification

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

Configuration Requirements: This test shall only be executed if IUT contains objects which will not accept a COV subscription. If every object in IUT will accept a COV subscription, then this test shall be skipped.

Test Steps:

1. TRANSMIT SubscribeCOV-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
2. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
   RECEIVE BACnet-Error PDU,
   'Error Class' = OBJECT,
   'Error Code' = OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED
   ELSE
   RECEIVE
   (BACnet-Error PDU,
   'Error Class' = OBJECT,
   'Error Code' = OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED) |
   (BACnet-Error PDU,
   Error Class = SERVICES,
   Error Code = SERVICE_REQUEST_DENIED | OTHER) |
   (BACnet-Error PDU,
   'Error Class' = PROPERTY,
   'Error Code' = NOT_COV_PROPERTY)

[In BTL Test Plan, modify reference for 9.10.2.1 from 135.1-2013 to BTL]

[In BTL Test Plan, modify the conditionality on test 9.10.2.1, as indicated]

4.10 Data Sharing - COV - B

4.10.1 Base Requirements

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

135.1-2013 - 9.10.1.1 - Confirmed COV Notifications		
	Test Method	Manual

	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	The functionality of this test is covered by tests 8.2.X which may be executed for each object type that supports COV. If the 8.2.X tests have been executed, then this test may be omitted.
	Test Directives	This test shall only be executed against objects which will accept the subscription.
	Testing Hints	
	Notes & Results	
. . .		
135.1-2013 - 9.10.2.1 - The Monitored Object Does Not Support COV Notification		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	This test shall only be executed if IUT contains objects which will not accept a COV subscription.~~Must be executed.~~
	Test Directives	
	Testing Hints	
	Notes & Results	

BTL-TP14.0b-3: Weaken Exception_Schedule length requirement

Overview:

In CR-0373, BTL-WG responded: "As there is no requirement in the standard for a length of the Exception_Schedule greater than 1, the BTL Test Package will be modified accordingly." This affects several tests, where a fixed length array was not anticipated in the language, as well as brings to light some errata.

Changes:

[In BTL Specified Tests, modify existing test 9.22.2.2]

9.22.2.2 Writing Array Properties with an Array Index that is Out of Range
Reason for Change: Modified test to remove dependency on EPICS values.

Purpose: To verify that the IUT can execute WriteProperty service requests when the requested property value is an array but the array index is out of range.

Test Concept: The TD shall select an object in the IUT that contains a writable array property designated P1. An attempt will be made to write to this property using an array index that is out of range. If no suitable object can be found, then this test shall be omitted.

Configuration Requirements: If the IUT supports any writable properties that are arrays, it shall be configured with at least one such property that can be used for this test.

Test Steps:

1. READ X = (Object1), P1

~~1. VERIFY (Object1), P1 = (the value defined for this property in the EPICS)~~

2. TRANSMIT WriteProperty-Request,
   'Object Identifier' = Object1,
   'Property Identifier' = P1,
   'Property Value' = (any value of the correct datatype for this property subject to the restrictions
   specified in the EPICS as defined in 4.4.2, except the value *X* read in step 1),
   'Property Array Index' = (any *value*~~*positive* integer~~ that is larger tha~~n~~t the *current*~~supported~~ size o~~f~~f the array)
3. RECEIVE BACnet-Error PDU,
   Error Class = PROPERTY,
   Error Code = INVALID_ARRAY_INDEX
4. VERIFY (Object1), P1 = *X*~~(the value defined for this property in the EPICS)~~

[In BTL Specified Tests, modify existing test 9.23.2.5]

9.23.2.5 Writing Array Properties with an Array Index that is Out of Range
Reason for Change: Modified test to remove dependency on EPICS values. Changed specification of the 'Property Array Index' in the request, since larger than the current size of the array, rather than the supported size, is the criterion that is the testable point.

Purpose: This test case verifies that the IUT can execute WritePropertyMultiple service requests when the requested property value is an array but the array index is out of range. This test shall only be performed if Protocol_Revision is present and has a value greater than or equal to 4.

Test Concept: The TD shall select an object, designated Object1, in the IUT that contains a writable array property designated P1. An attempt will be made to write to this property using an array index that is out of range. If no suitable object can be found, then this test shall be omitted.

Configuration Requirements: If the IUT supports any writable properties that are arrays, it shall be configured with at least one such property that can be used for this test.

Test Steps:

1. READ X = (Object1), P1

~~1. VERIFY (Object1), P1 = (the value defined for this property in the EPICS)~~
2. TRANSMIT WritePropertyMultiple-Request,
   'Object Identifier' = Object1,
   'Property Identifier' = P1,
   'Property Value' = (any *valid* value of the correct datatype for this property subject to the restrictions
   specified in the EPICS as defined in 4.4.2, except the value *X* read in step 1),
   `Property Array Index' = (any *value*~~positive integer~~ that is larger tha*n*~~t~~ the *current*~~supported~~ size of the array)
3. RECEIVE WritePropertyMultiple-Error,
   'Error Class' = PROPERTY,
   'Error Code' = INVALID_ARRAY_INDEX,
   'Object Identifier' = Object1,
   'Property Identifier' = P1
4. VERIFY (Object1), P1 = *X*~~(the value defined for this property in the EPICS)~~

[In BTL Test Plan, add a reference to test 9.23.2.5 in an existing section]

4.8 Data Sharing - WritePropertyMultiple - B

4.8.6 Contains Writable Array Properties

The IUT contains, or can be made to contain, a writeable array property.

135.1-2013 - 9.23.1.8 - Writing to Properties Based on Data Type		
	Test Method	Manual
	Configuration	The test shall be executed against a whole writable array property (no array index provided, and all elements written) in addition to configuration specified by ***ASHRAE 135.1-2013***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 9.23.2.5 - Writing Array Properties with an Array Index that is Out of Range		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

BTL-TP14.0b-4: Clarify WritePropertyMultiple Reject Test

Overview:

Existing test 9.23.2.X1 was misinterpreted at lab by our expression of the intent in the version that is in BTL Specified Tests-14.0. Also, when correctly applied, some IUT are observed making a response that is not among those specified.

Changes:

[In BTL Specified Tests, revise existing test 9.23.2.X1 as shown]

9.23.2.X1 WritePropertyMultiple Reject Test
Reason for Change: Addendum 135-2008u section 1.

Purpose: This test case verifies that the IUT does not send a Reject-PDU after applying part of a WritePropertyMultiple.

Test Concept: Two writable properties, P1 and P2 are written to the IUT but the portion of the WritePropertyMultiple specifying P2 is made invalid by omitting the 'Property Value' parameter. If the IUT returns a Reject, then the value of the first property is checked to ensure it has not changed.

Test Steps:

1. READ OldValue = O1, P1
2. TRANSMIT WritePropertyMultiple-Request,
   'Object Identifier' = O1,
   'Property Identifier' = P1,
   'Property Value' = (NewValue: any value other than OldValue that would be accepted by the IUT for P1)
   'Object Identifier' = O2,
   'Property Identifier' = P2
3. RECEIVEWritePropertyMultiple-Error,
   'Error Class' = SERVICES,
   'Error Code' = INVALID_TAG
   'Object Identifier' = O2
   'Property Identifier' = P2) |
   (RECEIVE BACnet-Reject-PDU,
   'Reject Reason' = INVALID_TAG | MISSING_REQUIRED_PARAMETER | INCONSISTENT_PARAMETERS | INVALID_PARAMETER_DATA_TYPE | TOO_MANY_ARGUMENTS)
4. IF (an Error-PDU was received in step 3) THEN
   VERIFY (O1), P1 = NewValue
   ELSE -- a Reject-PDU was received
   VERIFY (O1), P1 = OldValue

BTL-TP14.0b-5: Correct Trigger Verification Test

Overview:

The SSPC has adopted a different expression of this test as 7.3.2.24.18, without the correction to RECEIVE the last record with the LAST_ITEM flag set. BTL Specified Tests-14.0 had remedied that. This work-item further changes the range of the REPEAT, since the RECEIVE of the last record means the REPEAT loop should be one fewer.

Changes:
[In BTL Specified Tests: Modify existing test]

7.3.2.24.19 Trigger Verification Test

Reason for Change: This test has been included in 135.1-2013, but is here with a correction to the typo in Record_Count, with the steps renumbered to be consecutive, , with the distinct 'Result Flags' in the final record as noted in CR-0259, *the REPEAT loop should be one fewer,* and ~~with~~ the appropriate fields present in ReadRange-Request and ReadRange-ACK *are* based upon Record_Count, not Total_Record_Count, ~~to denote~~ *since this is* a request byPosition as ~~specifi~~*not*ed by CR-0282.

Purpose: To verify logged samples are based on the triggered Logging_Type.

Test Concept: The log, O_1 is configured to log based on TRIGGERED. Logging is enabled. After a period of time the buffer is checked to verify the data in the buffer is based on triggered values.

Configuration Requirements: The IUT shall be configured such that the monitored object's Logging_Type is set to TRIGGERED.

Test Steps:

1. WRITE Enable = FALSE
2. WRITE Record_Count = 0
3. WRITE Enable = TRUE
4. WAIT (10 seconds)
5. WRITE Trigger = TRUE
6. WAIT (20 seconds)
7. WRITE Trigger = TRUE
8. WAIT (40 seconds)
9. WRITE Trigger = TRUE
10. WAIT (30 seconds)
11. WRITE Enable = FALSE
12. READ N = Record_Count
13. REPEAT X = (1 through *34*)
    TRANSMIT ReadRange-*Request*
    'Object Identifier' = O_1,
    'Property Identifier' = Log_Buffer,
    'Reference Index' = N-4+X,
    'Count' = 1
    RECEIVE ~~ReadRangeAck~~*ReadRange-ACK*
    'Object Identifier' = O_1,
    'Property Identifier' = Log_Buffer,
    'Result Flags' = (False, False, False),
    'Item Count' = 1,
    'Item Data' = ~~(~~(one data record storing the timestamp in TS[X])~~)~~
14. *TRANSMIT ReadRange-Request*
    'Object Identifier' = O1,
    'Property Identifier' = Log_Buffer,
    'Reference Index' = N,
    'Count' = 1
    *RECEIVE ReadRange-*ACK~~ck~~
    'Object Identifier' = O1,
    'Property Identifier' = Log_Buffer,
    'Result Flags' = (False, True, False),

'Item Count' = 1,
'Item Data' = (one data record storing the timestamp in TS[4])

~~14~~*15*. CHECK(TS[2] - TS[1] ~= 20 seconds)
~~15~~*16*. CHECK(TS[3] - TS[2] ~= 40 seconds)
~~16~~*17*. CHECK(TS[4] - TS[3] ~= 30 seconds)

BTL-TP14.0b-6: Correct Unsupported Message Text Character Set AcknowledgeAlarm Test

Overview:

Test 9.1.X1 inadvertently specified the ACK_NOTIFICATION as an 'Event Type' that should be a 'Notify Type'.

Changes:
[In BTL Specified Tests, modify existing test 9.1.X1]

9.1.X1 Unsupported Message Text Character Set AcknowledgeAlarm Test

Reason for Change: Addendum 135-2010af added language to ensure that notifications are not ignored due to unsupported character sets.

Reference: 13.5.2

Purpose: To verify that the IUT does not fail to process an AcknowledgeAlarm request because the Acknowledgment Source parameter is of a character set that the IUT does not support.

Test Concept: Cause an event-initiating object, O1, in the IUT to transition to Event_State ES1. Acknowledge the transition and, in the AcknowledgeAlarm service, provide an 'Acknowledgment Source' parameter, AS1, which has a character set that the IUT does not support. Verify that the IUT processes the request even if the 'Acknowledgment Source' uses a character set that the IUT does not support, and that the IUT accepts and applies that Acknowledgment request, irrespective of the 'Acknowledgment Source'.

Configuration Requirements: Configure an event-initiating object, O1 which references a Notification Class object N1. Configure O1 such that it needs an acknowledgment when it transitions out of its current state. DELAY shall represent the time delay appropriate to the transition being tested (i.e. Time_Delay for to-offnormal, 0 for to-fault, and either Time_Delay or Time_Delay_Normal for to-normal). AS1 shall be a character string short enough for the IUT to receive and encoded in a character set that the IUT does not support. If the IUT supports all character sets, this test shall be skipped.

Test Steps:
1. MAKE(a condition exist which will cause O1 to transition)
2. WAIT DELAY
3. BEFORE **Notification Fail Time**
   RECEIVE UnconfirmedEventNotification-Request
   'Process Identifier' = (any valid process identifier),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = O1,
   'Time Stamp' = (TS1: any valid timestamp),
   Notification Class' = (N1: the Notification_Class configured in O1),
   'Priority' = (any valid priority),
   'Event Type' = (any standard event type),
   'Message Text' = (any valid text),
   'Notify Type' = ALARM | EVENT,
   'AckRequired' = TRUE,
   'From State' = (any valid event state),
   'To State' = (ES1: any valid event state),
   'Event Values' = (any values appropriate to the event type)
4. IF (ES1 = NORMAL) THEN
   VERIFY Acked_Transitions = (?,?,F)
   ELSE IF (ES1 = FAULT) THEN
   VERIFY Acked_Transitions = (?,F,?)
   ELSE
   VERIFY Acked_Transitions = (F,?,?)
5. TRANSMIT AcknowledgeAlarm-Request
   'Acknowledging Process Identifier' = (any valid value),
   'Event Object Identifier' = O1,
   'Event State Acknowledged' = ES1,
   'Time Stamp' = TS1,

'Acknowledgment Source' = AS1,
'Time of Acknowledgment' = (any valid timestamp)

6. RECEIVE BACnet-SimpleACK-PDU
7. BEFORE **Notification Fail Time**
RECEIVE UnconfirmedEventNotification-Request
'Process Identifier' = (any valid process identifier),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = O1,
'Time Stamp' = TS1
Notification Class' = (N1: the Notification_Class configured in O1),
'Priority' = (any valid priority),
~~'Event Type' = *(any valid event state)*ACK_NOTIFICATION,~~
'Event Type' = *(any standard event type)*~~ACK_NOTIFICATION~~,
'Message Text' = (any valid text),
'Notify Type' = *ACK_NOTIFICATION*~~ALARM | EVENT~~,
'To State' = ES1
8. IF (ES1 = NORMAL) THEN
VERIFY Acked_Transitions = (?,?,T)
ELSE IF (ES1 = FAULT) THEN
VERIFY Acked_Transitions = (?,T,?)
ELSE
VERIFY Acked_Transitions = (T,?,?)

Notes to Tester: The use of UnconfirmedEventNotification is specified in this test, solely to simplify the expression of the test. The behavior being tested applies to the ConfirmedEventNotification service as well.

BTL-TP14.0b-7: Weaken Enumerated type Requirement

Overview:

IC135-2012-DuffyO'Craven_SCHED-WS-I-B_BinaryPV_v1.doc, which was presented and discussed in Chicago in 2015, commented "While the BIBB is clear in what it requires, the committee agrees that this requirement should not be in the BIBB. Correspondingly, we make a change to lineitem “Is able to schedule Enumerated values” as an O instead of an R in our Test Package.

Changes:

[In BTL Checklist, change lineitem “Is able to schedule Enumerated values” to an O instead of an R and relocate it as indicated. Also reverse the mis-ordered footnotes 1 and 2]

Scheduling - Weekly Schedule - Internal - B		
	R~~2~~[1]	Base Requirements
	~~R~~	~~Is able to schedule Enumerated values~~
	R	Priority_For_Writing is writable
	R	Exception_Schedule property shall not be present
	C~~1~~[2]	Supports DM-TS-B
	C~~1~~[2]	Supports DM-UTC-B
	O	Supports configurable Effective_Period
	O	Is able to schedule values of different datatypes (not simultaneously)
	O	Is able to schedule NULL values
	O	Is able to schedule BOOLEAN values
	O	Is able to schedule Unsigned values
	O	Is able to schedule INTEGER (signed) values
	O	Is able to schedule REAL values
	O	Is able to schedule Double values
	O	Is able to schedule Octet String values
	O	Is able to schedule Character String values
	O	Is able to schedule Bit String values
	O	Is able to schedule Enumerated values
	O	Is able to schedule Date values
	O	Is able to schedule Time values
	O	Is able to schedule BACnetObjectIdentifier values
[1]This BIBB can only be claimed if SCHED-I-B is not claimed. [2]At least one of these options is required in order to claim conformance to this BIBB.		

[In BTL Test Plan, relocate formerly section 6.6.2 Is Able to Schedule Enumerated Values to before existing section 6.6.18 Is Able to Schedule Date Values, with all sections in between renumbered one lower as indicated.]

. . .

6.6 Schedule - Weekly Schedule - Internal - B

6.6.1Base Requirements

. . .

6.6.2~~3~~ Priority_For_Writing is Writable

The Priority_For_Writing property in the schedule object shall be writable..

. . .

6.6. 167 Is Able to Schedule Bit String Values

The IUT supports a Schedule object that is able to schedule (write) Bit String values.

. . .

6.6. 172 Is Able to Schedule Enumerated Values

The IUT supports a Schedule object that is able to schedule (write) Enumerated values.

. . .

6.6.18 Is Able to Schedule Date Values

The IUT supports a Schedule object that is able to schedule (write) Date values.

. . .