BACnet® TESTING LABORATORIES ADDENDA

Addendum fix1 to BTL Test Package 18.1

Revision 4
Revised 7/6/2021

Approved by the BTL Working Group on February 18, 2021;
Approved by the BTL Working Group Voting Members on July 7, 2021;
Published on July 13, 2021.

[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 18.1 is indicated through the use of *italics*, while deletions are indicated by ~~strikethrough~~. Where entirely new subclauses are proposed to be added, plain type is used throughout

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

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

BTL 18.1 fix1-1: Configurable Calendars (not writable) [BTLWG-391]

Overview:
Per BTL-CR-0429_Date_List_allowed_to_restrict_writability.doc; Clarify that Devices may be restricted and do not permit writing to the property 'Date_List', and the property may only be configured with a certain type of BACnetCalendarEntry e.g. BACnetDateRange.

Changes:

[In Test Plan sections 3.8.1 to modify Test Conditionality as shown]

3.8 Calendar Object

3.8.1 Base Requirements
Base requirements must be met by any IUT that can contain Calendar Objects

~~135.1-2019~~ *BTL* - 7.3.2.8.1 - Single Date Rollover Test		
	Test Conditionality	~~Must be executed.~~ *If the IUT does not contain any calendars with a writable Date_List and does not contain any calendars which can be configured with a Date entry, this test shall be skipped.*
	Test Directives	
	Testing Hints	
~~135.1-2019~~*BTL* - 7.3.2.8.2 - Date Range Test		
	Test Conditionality	~~Must be executed~~. *If the IUT does not contain any calendars with a writable Date_List and does not contain any calendars which can be configured with a BACnetDateRange entry, this test shall be skipped.*
	Test Directives	
	Testing Hints	
~~135.1-2019~~*BTL* - 7.3.2.8.3 - WeekNDay Test		
	Test Conditionality	~~Must be executed~~ *If the IUT does not contain any calendars with a writable Date_List and does not contain any calendars which can be configured with a BACnetWeekNDay entry, this test shall be skipped.*
	Test Directives	
	Testing Hints	
BTL - 7.2.X1 - Date Pattern Properties Test		
	Test Conditionality	~~Must be executed~~ *If the IUT does not contain any calendars with a writable Date_List and does not contain any calendars which can be configured with a BACnetCalendarEntry entry containing a Date, this test shall be skipped.*
	Test Directives	Apply to the Date_List property.
	Testing Hints	
BTL - 7.2.X7 - BACnetDateRange Non-Pattern Properties Test		
	Test Conditionality	This test shall only be applied to devices claiming Protocol_Revision 11 or higher. *If the IUT does not contain any calendars with a writable Date_List and does not contain any calendars which can be configured with a BACnetCalendarEntry entry containing a BACnetDateRange, this test shall be skipped.*
	Test Directives	Apply to Date_List property.
	Testing Hints	
BTL - 7.2.X8 - BACnetDateRange Open-Ended Pattern Properties Test		
	Test Conditionality	This test shall only be applied to devices claiming Protocol_Revision 11 or higher.

		If the IUT does not contain any calendars with a writable Date_List and does not contain any calendars which can be configured with a BACnetCalendarEntry entry containing a BACnetDateRange, this test shall be skipped.
	Test Directives	Apply to Date_List property.
	Testing Hints	
BTL - 9.23.2.X12 - BACnetDateRange Non-Pattern Properties Test using WritePropertyMultiple Service		
	Test Conditionality	This test shall only be applied to devices claiming Protocol_Revision 11 or higher and which supports execution of WritePropertyMultiple. *If the IUT does not contain any calendars with a writable Date_List and does not contain any calendars which can be configured with a BACnetCalendarEntry entry containing a BACnetDateRange, this test shall be skipped.*
	Test Directives	Apply to the Date_List property.
	Testing Hints	

[In BTL Specified Tests, add test 7.3.2.8.1 from 135.1-2019 and make the changes in the test cases as shown]

7.3.2.8.1 Single Date Rollover Test
Reason for Change: Allow for non-configurable Date_List properties.

Dependencies: ReadProperty Service Execution Tests, 9.18; TimeSynchronization Service Execution Tests, 9.30; UTCTimeSynchronization Service Execution Tests, 9.31.

BACnet Reference Clause: 12.9.

Purpose: To verify the ability to represent the Calendar status when the Date_List is in the form of an individual date. Either execution of the TimeSynchronization or the UTCTimeSynchronization service must be supported or another means must be supplied to reset the IUT's clock during the test.

Test Concept: The Calendar object is configured with a Date_List containing a single date. The IUT's clock is set to the date that immediately precedes the one specified in Date_List and a time near the end of the day. The test verifies that the Present_Value of the Calendar object is initially FALSE and that as the time rolls over to the next day the Present_Value changes to TRUE.

Configuration Requirements: The IUT shall be configured with a Calendar object that contains a Date_List with a single BACnetCalendarEntry in the form of a Date. *If Date_List property cannot be configured with a BACnetCalendarEntry in the form of a Date, then this test shall be skipped.*

Test Steps:
1. (TRANSMIT TimeSynchronization-Request,
'Time' = (the day preceding the one specified in Date_List,
24:00:00 – **Schedule Evaluation Fail Time** - 1 minute)) |
(TRANSMIT UTCTimeSynchronization-Request,
'Time' = (the day preceding the one specified in Date_List,
24:00:00 - **Schedule Evaluation Fail Time -** 1 minute converted to UTC)) |
MAKE (the local time = 24:00:00 - **Schedule Evaluation Fail Time -** 1 minute)
2. WAIT **Schedule Evaluation Fail Time**
3. VERIFY Present_Value = FALSE
4. WAIT (**Schedule Evaluation Fail Time** + 2 minutes)
5. VERIFY Present_Value = TRUE

[In BTL Specified Tests, add test 7.3.2.8.2 from 135.1-2019 and make the changes in the test cases as shown]

7.3.2.8.2 Date Range Test
Reason for Change: Allow for non-configurable Date_List properties.

Dependencies: ReadProperty Service Execution Tests, 9.18; TimeSynchronization Service Execution Tests, 9.30; UTCTimeSynchronization Service Execution Tests, 9.31.

BACnet Reference Clause: 12.9.

Purpose: To verify the ability to represent the Calendar status when the Date_List is in the form of a BACnetDateRange. Either execution of the TimeSynchronization or the UTCTimeSynchronization service must be supported or another means must be supplied to reset the IUT's clock during the test.

Test Concept: The Calendar object is configured with a Date_List containing a single BACnetDateRange. The IUT's clock is set to a time and date that is outside of the date range. The Present_Value is read and verified to be FALSE. The clock is reset to a value within the date range and the Present_Value is read again to verify that it has the value TRUE. If the IUT can be configured with wildcard fields in the date range then it shall be tested with and without wildcards.

Configuration Requirements: The IUT shall be configured with a Calendar object that contains a Date_List with a single BACnetCalendarEntry in the form of a BACnetDateRange.-*If Date_List property cannot be configured with a BACnetCalendarEntry in the form of a BACnetDateRange, then this test shall be skipped.*

Test Steps:
1. (TRANSMIT TimeSynchronization-Request,
   'Time' = (any day and time outside of the specified date range selected by the tester)) |
   (TRANSMIT UTCTimeSynchronization-Request,
   'Time' = (any day and time outside of the specified date range selected by the tester)) |
   MAKE (the local time = any day and time outside of the specified date range selected by the tester)
2. WAIT **Schedule Evaluation Fail Time**
3. VERIFY Present_Value = FALSE
4. (TRANSMIT TimeSynchronization-Request,
   'Time' = (any day and time inside the specified date range selected by the tester)) |
   (TRANSMIT UTCTimeSynchronization-Request,
   'Time' = (any day and time inside the specified date range selected by the tester)) |
   MAKE (the local time = any day and time inside the specified date range selected by the tester)
5. WAIT **Schedule Evaluation Fail Time**
6. VERIFY Present_Value = TRUE

[In BTL Specified Tests, add test 7.3.2.8.3 from 135.1-2019 and make the changes in the test cases as shown]

7.3.2.8.3 WeekNDay Test
Reason for Change: Allow for non-configurable Date_List properties.

Dependencies: ReadProperty Service Execution Tests, 9.18; TimeSynchronization Service Execution Tests, 9.30; UTCTimeSynchronization Service Execution Tests, 9.31.

BACnet Reference Clause: 12.9.

Purpose: To verify the ability to represent the Calendar status when the Date_List is in the form of a BACnetWeekNDay. Either execution of the TimeSynchronization or the UTCTimeSynchronization service must be supported or another means must be supplied to reset the IUT's clock during the test.

Test Concept: The Calendar object is configured with a Date_List containing a single BACnetWeekNDay. The IUT's clock is set to a time and date, T1, that matches the BACnetWeekNDay mask. The Present_Value is read and verified to be TRUE. The clock is reset to a value, T2, that matches the BACnetWeekNDay mask except for the month. The Present_Value is read and verified to be FALSE. The clock is reset again to a value, T3, that matches the BACnetWeekNDay mask except for the week of the month. The Present_Value is read and verified to be FALSE. The clock is reset again to a value, T4, that matches the BACnetWeekNDay mask except for the day of the week. The Present_Value is read and verified to be FALSE. In between each change, the clock is reset to T1 to force the Present_Value back to TRUE.

Configuration Requirements: The IUT shall be configured with a Calendar object that contains a Date_List with a single BACnetCalendarEntry in the form of a BACnetWeekNDay. The BACnetWeekNDay shall be the 11th month, last seven

days, and Saturday. *If Date_List property cannot be configured with a BACnetCalendarEntry in the form of a BACnetWeekNDay, then this test shall be skipped.*

Test Steps:

1. (TRANSMIT TimeSynchronization-Request,
   'Time' = (T1) |
   (TRANSMIT UTCTimeSynchronization-Request,
   'Time' = (T1)) |
   MAKE (the local time = T1)
2. WAIT **Schedule Evaluation Fail Time**
3. VERIFY Present_Value = TRUE
4. (TRANSMIT TimeSynchronization-Request,
   'Time' = (T2) |
   (TRANSMIT UTCTimeSynchronization-Request,
   'Time' = (T2)) |
   MAKE (the local time = T2)
5. WAIT **Schedule Evaluation Fail Time**
6. VERIFY Present_Value = FALSE
7. (TRANSMIT TimeSynchronization-Request,
   'Time' = (T1)) |
   (TRANSMIT UTCTimeSynchronization-Request,
   'Time' = (T1)) |
   MAKE (the local time = T1)
8. WAIT **Schedule Evaluation Fail Time**
9. VERIFY Present_Value = TRUE
10. (TRANSMIT TimeSynchronization-Request,
    'Time' = (T3) |
    (TRANSMIT UTCTimeSynchronization-Request,
    'Time' = (T3)) |
    MAKE (the local time = T3)
11. WAIT **Schedule Evaluation Fail Time**
12. VERIFY Present_Value = FALSE
13. (TRANSMIT TimeSynchronization-Request,
    'Time' = (T1)) |
    (TRANSMIT UTCTimeSynchronization-Request,
    'Time' = (T1)) |
    MAKE (the local time = T1)
14. WAIT **Schedule Evaluation Fail Time**
15. VERIFY Present_Value = TRUE
16. (TRANSMIT TimeSynchronization-Request,
    'Time' = (T4) |
    (TRANSMIT UTCTimeSynchronization-Request,
    'Time' = (T4)) |
    MAKE (the local time = T4)
17. WAIT **Schedule Evaluation Fail Time**
18. VERIFY Present_Value = FALSE

7.2.X1 Date Pattern Properties Test

Reason for Change: Addendum 135-2001a-1 adds odd and even month support, and last-day-of-the-month special value. Addendum 135-2008h-8 adds odd and even day support. Addendum 135-2008ac-1 clarifies when wildcards are allowed in dates and times. Test does not exist in 135.1-2019. *Allow for non-configurable Date_List properties.*

Purpose: To verify that the property being tested accepts special date field values.

Test Concept: The property being tested, P1, is written with each of the special date field values to ensure that the property accepts them. A date, D1, is selected which is within the date range that the IUT will accept for the property. The value, written to the property is the date D1 with one of its fields replaced with one of the date special values. If the property is a

complex datatype, the other fields in the value shall be set within the range accepted by the IUT. The list of Specials comes from the Chapter 21 Application Types section on Date. The day-of-week field is redundant information and can be regenerated from the other fields. In order to not fail devices which always ensure this field is consistent with the other fields in the date value, the use of an unspecified day of week is always tested in conjunction with another pattern value.

Configuration Requirements: The IUT shall be configured with a Calendar object that contains a Date_List with a single BACnetCalendarEntry in the form of a Date. If Date_List property cannot be configured with a BACnetCalendarEntry in the form of a Date, then this test shall be skipped.

Notes to Tester: if P1 is an array, then an array index shall be provided in the WRITE and VERIFY operations.

Test Steps:

1. IF (Protocol_Revision is not present or Protocol_Revision < 4) *THEN*
   Specials = (year unspecified, month unspecified, day of month unspecified)
   ELSE IF (Protocol_Revision >= 4 and Protocol_Revision < 10) *THEN*
   Specials = (year unspecified, month unspecified, day of month unspecified, odd months, even months, last day of month)
   ELSE
   Specials = (year unspecified, month unspecified, day of month unspecified, odd months, even months, last day of month, even days, odd days)
2. REPEAT SV = (each value in Specials) DO {
   IF SV <> day of week unspecified THEN
   V1 = D1 updated with the value SV
   ELSE
   V1 = D1 updated with the value SV and any other value from Specials
   WRITE P1 = (V1)
   VERIFY P1 = (V1)
   }

7.2.X7 BACnetDateRange Non-Pattern Properties Test

Reason for Change: Addendum 135-2008ac-1 clarifies in the clause 12 preamble, when wildcards are allowed in BACnetDateRange. *Allow for non-configurable Date_List properties.*

Purpose: To verify that the property being tested does not accept special date field values.

Test Concept: A BACnetDateRange property, or property that is a complex datatype containing a BACnetDateRange, P1 is written with each of the special date field values to ensure that the property does not accept them. Each half of the dateRange DR1 is selected so it is within the range that the IUT will accept for the property. The value, V1, written to the property is the dateRange DR1 with one of its fields replaced with one of the date special values. If the property is a complex datatype such as a BACnetCalenderEntry, the other fields in the value shall be set within the range accepted by the IUT.

Configuration Requirements: This test shall only be applied to devices claiming Protocol_Revision 11 or higher. *The IUT shall be configured with a Calendar object that contains a Date_List with a single BACnetCalendarEntry in the form of a BACnetDateRange. If Date_List property cannot be configured with a BACnetCalendarEntry in the form of a BACnetDateRange, then this test shall be skipped.*

Notes to Tester: if P1 is an array, then an array index shall be provided in the TRANSMIT portion of step 1.

Test Steps:

1. REPEAT SV = (year unspecified, month unspecified, day of month unspecified, odd months, even months, last day of month, even days, odd days) DO {
   TRANSMIT WriteProperty-Request
   'Object Identifier' = O1,
   'Property Identifier' = P1,
   'Property Value' = (DR1 with startDate updated with the special value SV)
   RECEIVE BACnet-Error-PDU

'Error Class' = PROPERTY,
'Error Code' = VALUE_OUT_OF_RANGE
TRANSMIT WriteProperty-Request
'Object Identifier' = O1,
'Property Identifier' = P1,
'Property Value' = (DR1 with endDate updated with the special value SV)
Receive BACnet-Error-PDU
'Error Class' = PROPERTY,
'Error Code' = VALUE_OUT_OF_RANGE

7.2.X8 BACnetDateRange Open-Ended Pattern Properties Test

Reason for Change: Addendum 135-2008ac-1 clarifies in the clause 12 preamble, when wildcards are allowed in BACnetDateRange. *Allow for non-configurable Date_List properties.*

Purpose: To verify that the property being tested accepts a fully unspecified date in either or both halves of the value.

Test Concept: A BACnetDateRange property, or property that has a complex datatype containing a BACnetDateRange, P1 is written with a fully unspecified date in either or both halves to ensure that the property accepts them. DR1 is selected which is within the date range that the IUT will accept for the property. The value, written to the property is the dateRange DR1 replaced with a fully unspecified date in either or both startDate and endDate. If the property is a complex datatype the other fields in the value shall be set within the range accepted by the IUT.

Configuration Requirements: This test shall only be applied to devices claiming Protocol_Revision 11 or higher. *The IUT shall be configured with a Calendar object that contains a Date_List with a single BACnetCalendarEntry in the form of a BACnetDateRange.If Date_List property cannot be configured with a BACnetCalendarEntry in the form of a BACnetDateRange, then this test shall be skipped.*

Notes to Tester: if P1 is an array, then an array index shall be provided in the WRITEs and VERIFYs.

Test Steps:

1. WRITE P1 = (DR1 updated with a fully unspecified date in startDate)
2. VERIFY P1 = (the value written)
3. WRITE P1 = (DR1 updated with a fully unspecified date in endDate)
4. VERIFY P1 = (the value written)
5. WRITE P1 = (DR1 updated with a fully unspecified date in both startDate and endDate)
6. VERIFY P1 = (the value written)

BTL 18.1-fix1-2: Update Alarming Tests for NORMAL to NORMAL Transitions [BTLWG-843]

Overview:
CR-0382 response specified that all tests in section 9.1.1and 9.1.2 to include ToFault transitions and application of the tests in NORMAL to NORMAL Transitions. Test 9.1.1.8 is used as example.

Changes:

[In BTL Test Plan, modify all references to tests:, 9.1.1.4, 9.1.1.10, 9.1.1.11, 9.1.2.1, 9.1.2.5 from 135.1-2019 to BTL]

[In BTL Specified Tests, modify tests 9.1.1.1]

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

Reason For Change: Made changes to allow cases where only one Recipient_List entry is supported. *Made changes to include not supported transitions.*

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

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

Configuration Requirements: The IUT shall be configured with ~~at object~~ *one object* that can detect alarm conditions and send confirmed notifications. The Acked_Transitions property shall have the value B'111' indicating that all transitions have been acknowledged. The TD and one other BACnet device, if the IUT supports multiple recipients, shall be recipients of the alarm notification. *D1 is either the pTimeDelay, or pTimeDelayNormal parameter, or 0 (for transitions to and from FAULT state) depending on the event transition.*

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

Test Steps:

1. MAKE (a change that triggers the detection of an alarm event in the IUT)
2. WAIT (~~pTimeDelay~~*D1*)
3. BEFORE **Notification Fail Time**
   RECEIVE ConfirmedEventNotification-Request,
   'Process Identifier' = (the process identifier configured for this event),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (the object detecting the alarm),
   'Time Stamp' = (T1: any valid time stamp),
   'Notification Class' = (the notification class configured for this event),
   'Priority' = (the priority configured for this event),
   'Event Type' = (any valid event type),
   'Message Text' = (optional, any valid message text),
   'Notify Type' = (the notify type configured for this event),
   'AckRequired' = TRUE,
   'From State' = ~~NORMAL~~*(any appropriate event state)*,
   'To State' = (any appropriate ~~non-normal~~ event state),
   'Event Values' = (the values appropriate to the event type)
4. TRANSMIT BACnet-SimpleACK-PDU
5. RECEIVE
   DESTINATION = (a device other than the TD),

SOURCE = IUT,
ConfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (T1: any valid time stamp),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event),
'Event Type' = (any valid event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = (the notify type configured for this event),
'AckRequired' = TRUE,
'From State' = ~~NORMAL~~*(any appropriate event state)*,
'To State' = (any appropriate ~~non-normal~~ event state),
'Event Values' = (the values appropriate to the event type)

6. TRANSMIT BACnet-SimpleACK-PDU
7. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = ~~(FALSE, TRUE, TRUE)~~*(appropriate bit FALSE, the others TRUE)*
8. TRANSMIT AcknowledgeAlarm-Request,
   'Acknowledging Process Identifier' = (the value of the 'Process Identifier' parameter in the event notification),
   'Event Object Identifier' = (the 'Event Object Identifier' from the event notification),
   'Event State Acknowledged' = (the state specified in the 'To State' parameter of the notification),
   'Time Stamp' = (the time stamp conveyed in the notification),
   'Acknowledgement Source' = (any valid value),
   'Time of Acknowledgment' = (the TD's current time using a Time format)
9. RECEIVE BACnet-Simple-ACK-PDU
10. IF (Protocol_Revision is present and Protocol_Revision ≥ 1) THEN
    BEFORE **Notification Fail Time**
    RECEIVE ConfirmedEventNotification-Request,
    'Process Identifier' = (the process identifier configured for this event),
    'Initiating Device Identifier' = IUT,
    'Event Object Identifier' = (the object detecting the alarm),
    'Time Stamp' = (any valid time stamp),
    'Notification Class' = (the notification class configured for this event),
    'Priority' = (the priority configured for this event),
    'Event Type' = (the event type included in step 3),
    'Message Type' = (optional, any valid message text),
    'Notify Type' = ACK_NOTIFICATION,
    'To State' = (the 'To State' used in step 3)
    ELSE
    BEFORE **Notification Fail Time**
    RECEIVE ConfirmedEventNotification-Request,
    'Process Identifier' = (the process identifier configured for this event),
    'Initiating Device Identifier' = IUT,
    'Event Object Identifier' = (the object detecting the alarm),
    'Time Stamp' = (any valid time stamp),
    'Notification Class' = (the notification class configured for this event),
    'Priority' = (the priority configured for this event),
    'Event Type' = (the event type included in step 3),
    'Message Type' = (optional, any valid message text),
    'Notify Type' = ACK_NOTIFICATION,
11. TRANSMIT BACnet-SimpleACK-PDU
12. IF (Protocol_Revision is present and Protocol_Revision ≥ 1) THEN
    BEFORE **Notification Fail Time**
    RECEIVE
    DESTINATION = (a device other than the TD),
    SOURCE = IUT,
    ConfirmedEventNotification-Request,
    'Process Identifier' = (the process identifier configured for this event),

'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (the timestamp or sequence number received in step 10),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event),
'Event Type' = (the event type included in step 3),
'Message Type' = (optional, any valid message text),
'Notify Type' = ACK_NOTIFICATION,
'To State' = (the 'To State' used in step 3)

ELSE
BEFORE **Notification Fail Time**
RECEIVE
DESTINATION = (a device other than the TD),
SOURCE = IUT,
ConfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (the timestamp or sequence number received in step 10),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event),
'Event Type' = (the event type included in step 3),
'Message Type' = (optional, any valid message text),
'Notify Type' = ACK_NOTIFICATION

13. TRANSMIT BACnet-SimpleACK-PDU
14. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = (TRUE,TRUE,TRUE)

[In BTL Specified Tests, add tests 9.1.1.4, 9.1.1.10, 9.1.1.11, 9.1.2.1, 9.1.2.5 from 135.1-2019 and modify as noted.]

9.1.1.4 Successful Alarm Acknowledgment of Unconfirmed Event Notifications Using the Time Form of the 'Time of Acknowledgment' Parameter

Reason For Change: *Made changes to include not supported transitions.*

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

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

Configuration Requirements: The IUT shall be configured with at least one object that can detect alarm conditions and send unconfirmed notifications. The Acked_Transitions property shall have the value B'111' indicating that all transitions have been acknowledged. The TD and one other BACnet device, if the IUT supports multiple recipients, shall be recipients of the alarm notification. *D1 is either the pTimeDelay, or pTimeDelayNormal parameter, or 0 (for transitions to and from FAULT state) depending on the event transition.*

Notes to Tester: The destination address used for the acknowledgment notification in step 8 shall be the same address used in step 3. The destination address used for the acknowledgment notification in step 9 shall be the same address used in step 4. Inclusion of the 'To State' parameter in acknowledgement notifications was added in protocol version 1, protocol revision 1. Implementations that precede this version will not include this parameter. When multiple event notifications are expected for a specific event, the order that the IUT transmits them in is irrelevant. *If the IUT can only be configured with one recipient in the Recipient_List property of the issuing Notification_class object, omit steps 4 and 9.*

Test Steps:
1. MAKE (a change that triggers the detection of an alarm event in the IUT)
2. WAIT (~~pTimeDelay~~*D1*)
3. BEFORE **Notification Fail Time**
   RECEIVE,

DESTINATION = LOCAL BROADCAST | GLOBAL BROADCAST | TD,
SOURCE = IUT,
UnconfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (any valid time stamp),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event type),
'Event Type' = (any valid event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = (the notify type configured for this event),
'AckRequired' = TRUE,
'From State' = ~~NORMAL~~*(any appropriate event state)*,
'To State' = (any appropriate event state),
'Event Values' = (the values appropriate to the event type)

4. IF (the notification in step 3 was not a broadcast) THEN
RECEIVE
DESTINATION = (a device other than the TD),
SOURCE = IUT,
UnconfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (the timestamp or sequence number received in step 3),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event type),
'Event Type' = (any valid event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = (the notify type configured for this event),
'AckRequired' = TRUE,
'From State' = ~~NORMAL~~*(any appropriate event state)*,
'To State' = (any appropriate event state),
'Event Values' = (the values appropriate to the event type)
5. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = ~~(FALSE,TRUE,TRUE)~~(appropriate bit FALSE, the others TRUE)
6. TRANSMIT AcknowledgeAlarm-Request,
'Acknowledging Process Identifier' = (the value of the 'Process Identifier' parameter in the event notification),
'Event Object Identifier' = (the 'Event Object Identifier' from the event notification),
'Event State Acknowledged' = (the state specified in the 'To State' parameter of the notification),
'Time Stamp' = (any valid time stamp),
'Acknowledgement Source' = (any valid value),
'Time of Acknowledgment' = (the TD's current time using a Time format)
7. RECEIVE BACnet-Simple-ACK-PDU
8. IF (Protocol_Revision is present AND Protocol_Revision ≥ 1) THEN
BEFORE **Notification Fail Time**
RECEIVE
DESTINATION = LOCAL BROADCAST | GLOBAL BROADCAST | TD,
SOURCE = IUT,
UnconfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (any valid time stamp),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event type),
'Event Type' = (any valid event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = ACK_NOTIFICATION,
'To State' = (the 'To State' used in step 3 or 4)

ELSE
BEFORE **Notification Fail Time**
RECEIVE
DESTINATION = LOCAL BROADCAST | GLOBAL BROADCAST | TD,
SOURCE = IUT,
UnconfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (the current time or sequence number),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event type),
'Event Type' = (any valid event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = ACK_NOTIFICATION

9. IF (the notification in step 8 was not broadcast) THEN
IF (Protocol_Revision is present AND Protocol_Revision ≥ 1) THEN
RECEIVE
DESTINATION = (at least one device other than the TD),
SOURCE = IUT,
UnconfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (the timestamp or sequence number from the notification in step 8),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event type),
'Event Type' = (any valid event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = ACK_NOTIFICATION,
'To State' = (the 'To State' used in step 3 or 4)
ELSE
RECEIVE
DESTINATION = (at least one device other than the TD),
SOURCE = IUT,
UnconfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (the timestamp or sequence number from the notification in step 8),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event type),
'Event Type' = (any valid event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = ACK_NOTIFICATION,
10. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = (TRUE,TRUE,TRUE)

~~Notes to Tester: The destination address used for the acknowledgment notification in step 8 shall be the same address used in step 3. The destination address used for the acknowledgment notification in step 9 shall be the same address used in step 4. Inclusion of the 'To State' parameter in acknowledgement notifications was added in protocol version 1, protocol revision 1. Implementations that precede this version will not include this parameter. When multiple event notifications are expected for a specific event, the order that the IUT transmits them in is irrelevant. *If the IUT can only be configured with one recipient in the Recipient_List property of the issuing Notification_class object, omit steps 4 and 9.*~~

9.1.1.10 Successful Alarm Re-Acknowledgment of Confirmed Event Notifications

Reason For Change: Made changes to include not supported transitions.

Purpose: To verify the successful re-acknowledgment of an event signaled by a ConfirmedEventNotification, including notification of other workstations and updating of the Acked_Transitions status.

Test Concept: An event is triggered and the IUT notifies the TD and one other device. The TD acknowledges the event and verifies that the acknowledgment is properly noted by the IUT. The IUT notifies all recipients that the event has been acknowledged. The TD then acknowledges the event again, and the IUT again notifies all recipients. This behavior was not defined before Protocol_Revision 7 and so this test shall only be performed if Protocol_Revision is present (i.e., Protocol_Revision ≥ 7).

Configuration Requirements: The IUT shall be configured with at least one event-initiating object that generates offnormal transitions and sends confirmed notifications. The Acked_Transitions property shall have the value B'111', indicating that all transitions have been acknowledged. The TD and one other BACnet device, if the IUT supports multiple recipients, shall be recipients of the event notification. *D1 is either the pTimeDelay, or pTimeDelayNormal parameter, or 0 (for transitions to and from FAULT state) depending on the event transition.*

Notes to Tester: The destination address used for the acknowledgment notification in steps 12 and 19 shall be the same address used in step 4. When multiple event notifications are expected for a specific event, the order that the IUT transmits them in is irrelevant. If the IUT can only be configured with one recipient in the Recipient_List property of the issuing Notification_class object, omit steps 5, 6, 12, 13, 19, and 20.

Test Steps:

1. MAKE (a change that triggers the detection of an event in the IUT)
2. WAIT (~~pTimeDelay~~*D1*)
3. BEFORE **Notification Fail Time**
   RECEIVE ConfirmedEventNotification-Request,
   'Process Identifier' = (the process identifier configured for this event),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (the event-initiating object),
   'Time Stamp' = (T1: any valid time stamp),
   'Notification Class' = (the notification class configured for this event),
   'Priority' = (the priority configured for this event),
   'Event Type' = (E1: any valid event type),
   'Message Text' = (optional, any valid message text),
   'Notify Type' = (the notify type configured for this event),
   'AckRequired' = TRUE,
   'From State' = ~~NORMAL~~*(any appropriate event state)*,
   'To State' = (S1: any appropriate ~~offnormal~~ event state),
   'Event Values' = (the values appropriate to the event type)
4. TRANSMIT BACnet-SimpleACK-PDU
5. RECEIVE
   DESTINATION = (a device other than the TD),
   SOURCE = IUT,
   ConfirmedEventNotification-Request,
   'Process Identifier' = (the process identifier configured for this event),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (the event-initiating object),
   'Time Stamp' = (T1),
   'Notification Class' = (the notification class configured for this event),
   'Priority' = (the priority configured for this event),
   'Event Type' = (any valid event type),
   'Message Text' = (optional, any valid message text),
   'Notify Type' = (the notify type configured for this event),
   'AckRequired' = TRUE,
   'From State' = ~~NORMAL~~*(any appropriate event state)*,
   'To State' = (any appropriate ~~offnormal~~ event state),
   'Event Values' = (the values appropriate to the event type)
6. TRANSMIT BACnet-SimpleACK-PDU
7. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = ~~(FALSE,TRUE,TRUE)~~ (appropriate bit FALSE, the others TRUE)

8. TRANSMIT AcknowledgeAlarm-Request,
   'Acknowledging Process Identifier' = (the value of the 'Process Identifier' parameter in the event notification),
   'Event Object Identifier' = (the 'Event Object Identifier' from the event notification),
   'Event State Acknowledged' = (the state specified in the 'To State' parameter of the notification),
   'Time Stamp' = (any valid time stamp),
   'Acknowledgment Source' = (any valid value)
   'Time of Acknowledgment' = (any of the forms specified for this parameter)
9. RECEIVE BACnet-Simple-ACK-PDU
10. BEFORE **Notification Fail Time**
    RECEIVE ConfirmedEventNotification-Request,
    'Process Identifier' = (the process identifier configured for this event),
    'Initiating Device Identifier' = IUT,
    'Event Object Identifier' = (the event-initiating object),
    'Time Stamp' = (T~~1~~*2*: any valid time stamp),
    'Notification Class' = (the notification class configured for this event),
    'Priority' = (the priority configured for this event),
    'Event Type' = (E1),
    'Message Text' = (optional, any valid message text),
    'Notify Type' = ACK_NOTIFICATION,
    'To State' = (S1)
11. TRANSMIT BACnet-SimpleACK-PDU
12. RECEIVE
    DESTINATION = (a device other than the TD),
    SOURCE = IUT,
    ConfirmedEventNotification-Request,
    'Process Identifier' = (the process identifier configured for this event),
    'Initiating Device Identifier' = IUT,
    'Event Object Identifier' = (the event-initiating object),
    'Time Stamp' = (T~~1~~*2*),
    'Notification Class' = (the notification class configured for this event),
    'Priority' = (the priority configured for this event),
    'Event Type' = (E1),
    'Message Text' = (optional, any valid message text),
    'Notify Type' = ACK_NOTIFICATION,
    'To State' = (S1)
13. TRANSMIT BACnet-SimpleACK-PDU
14. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = (TRUE, TRUE, TRUE)
15. TRANSMIT AcknowledgeAlarm-Request,
    'Acknowledging Process Identifier' = (the value of the 'Process Identifier' parameter in the event notification),
    'Event Object Identifier' = (the 'Event Object Identifier' from the event notification),
    'Event State Acknowledged' = (the state specified in the 'To State' parameter of the notification),
    'Time Stamp' = (any valid time stamp),
    'Acknowledgment Source' = (any valid value)
    'Time of Acknowledgment' = (any of the forms specified for this parameter)
16. RECEIVE BACnet-SimpleACK-PDU
17. BEFORE **Notification Fail Time**
    RECEIVE ConfirmedEventNotification-Request,
    'Process Identifier' = (the process identifier configured for this event),
    'Initiating Device Identifier' = IUT,
    'Event Object Identifier' = (the event-initiating object),
    'Time Stamp' = (T~~2~~*3*: any valid time stamp),
    'Notification Class' = (the notification class configured for this event),
    'Priority' = (the priority configured for this event),
    'Event Type' = (E1),
    'Message Text' = (optional, any valid message text),
    'Notify Type' = ACK_NOTIFICATION,
    'To State' = (S1)
18. TRANSMIT BACnet-SimpleACK-PDU
19. RECEIVE
    DESTINATION = (at least one device other than the TD),

SOURCE = IUT,
ConfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the event-initiating object),
'Time Stamp' = (T~~2~~*3*),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event),
'Event Type' = (E1),
'Message Text' = (optional, any valid message text),
'Notify Type' = ACK_NOTIFICATION,
'To State' = (S1)

20. TRANSMIT BACnet-SimpleACK-PDU
21. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = (TRUE, TRUE, TRUE)

~~Notes to Tester: The destination address used for the acknowledgment notification in steps 12 and 19 shall be the same address used in step 4. When multiple event notifications are expected for a specific event, the order that the IUT transmits them in is irrelevant. If the IUT can only be configured with one recipient in the Recipient_List property of the issuing Notification_class object, omit steps 5, 6, 12, 13, 19, and 20.~~

9.1.1.11 Successful Alarm Re-Acknowledgment of Unconfirmed Event Notifications

Reason For Change: Made changes to include not supported transitions.

Purpose: To verify the successful re-acknowledgment of an event signaled by an UnconfirmedEventNotification, including notification of other workstations and updating of the Acked_Transitions status.

Test Concept: An event is triggered and the IUT notifies the TD and one other device. The TD acknowledges the event and verifies that the acknowledgment is properly noted by the IUT. The IUT notifies all recipients that the event has been acknowledged. The TD then acknowledges the event again, and the IUT again notifies all recipients. This behavior was not defined before Protocol_Revision 7 and so this test shall only be performed if Protocol_Revision is present (i.e., Protocol_Revision ≥ 7).

Configuration Requirements: The IUT shall be configured with at least one event-initiating object that generates offnormal transitions and sends unconfirmed notifications. The Acked_Transitions property shall have the value B'111', indicating that all transitions have been acknowledged. The TD and one other BACnet device, if the IUT supports multiple recipients, shall be recipients of the event notification. *D1 is either the pTimeDelay, or pTimeDelayNormal parameter, or 0 (for transitions to and from FAULT state) depending on the event transition.*

Notes to Tester: The destination address used for the acknowledgment notification in steps 9 and 14 shall be the same address used in step 4. When multiple event notifications are expected for a specific event, the order that the IUT transmits them in is irrelevant. If the IUT can only be configured with one recipient in the Recipient_List property of the issuing Notification_class object, omit steps 4, 9, and 14.

Test Steps:

1. MAKE (a change that triggers the detection of an offnormal event in the IUT)
2. WAIT (~~pTimeDelay~~*D1*)
3. BEFORE **Notification Fail Time**
   RECEIVE UnconfirmedEventNotification-Request,
   'Process Identifier' = (the process identifier configured for this event),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (the event-initiating object),
   'Time Stamp' = (T1: any valid time stamp),
   'Notification Class' = (the notification class configured for this event),
   'Priority' = (the priority configured for this event),
   'Event Type' = (E1: any valid event type),
   'Message Text' = (optional, any valid message text),
   'Notify Type' = (the notify type configured for this event),

'AckRequired' = TRUE,
'From State' = ~~NORMAL~~*(any appropriate event state)*,
'To State' = (S1: any appropriate ~~offnormal~~ event state),
'Event Values' = (the values appropriate to the event type)

4. RECEIVE
DESTINATION = (at least one device other than the TD),
SOURCE = IUT,
UnconfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the event-initiating object),
'Time Stamp' = (T1),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event),
'Event Type' = (any valid event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = (the notify type configured for this event),
'AckRequired' = TRUE,
'From State' = ~~NORMAL~~*(any appropriate event state)*,
'To State' = (any appropriate ~~offnormal~~ event state),
'Event Values' = (the values appropriate to the event type)
5. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = ~~(FALSE, TRUE, TRUE)~~*(*appropriate bit FALSE, the others TRUE*)*
6. TRANSMIT AcknowledgeAlarm-Request,
'Acknowledging Process Identifier' = (the value of the 'Process Identifier' parameter in the event notification),
'Event Object Identifier' = (the 'Event Object Identifier' from the event notification),
'Event State Acknowledged' = (the state specified in the 'To State' parameter of the notification),
'Time Stamp' = (any valid time stamp),
'Acknowledgment Source' = (any valid value)
'Time of Acknowledgment' = (any of the forms specified for this parameter)
7. RECEIVE BACnet-SimpleACK-PDU
8. BEFORE **Notification Fail Time**
RECEIVE UnconfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the event-initiating object),
'Time Stamp' = (T2: any valid time stamp),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event),
'Event Type' = (E1),
'Message Text' = (optional, any valid message text),
'Notify Type' = ACK_NOTIFICATION,
'To State' = (S1)
9. RECEIVE
DESTINATION = (a device other than the TD),
SOURCE = IUT,
UnconfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the event-initiating object),
'Time Stamp' = (T2),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event),
'Event Type' = (E1),
'Message Text' = (optional, any valid message text),
'Notify Type' = ACK_NOTIFICATION,
'To State' = (S1)
10. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = (TRUE, TRUE, TRUE)
11. TRANSMIT AcknowledgeAlarm-Request,
'Acknowledging Process Identifier' = (the value of the 'Process Identifier' parameter in the event notification),

'Event Object Identifier' = (the 'Event Object Identifier' from the event notification),
'Event State Acknowledged' = (the state specified in the 'To State' parameter of the notification),
'Time Stamp' = (the time stamp conveyed in the notification),
'Acknowledgment Source' = (any valid value)
'Time of Acknowledgment' = (any of the forms specified for this parameter)

12. RECEIVE BACnet-SimpleACK-PDU
13. BEFORE **Notification Fail Time**
    RECEIVE UnconfirmedEventNotification-Request,
    'Process Identifier' = (the process identifier configured for this event),
    'Initiating Device Identifier' = IUT,
    'Event Object Identifier' = (the event-initiating object),
    'Time Stamp' = (T3: any valid time stamp),
    'Notification Class' = (the notification class configured for this event),
    'Priority' = (the priority configured for this event),
    'Event Type' = (E1),
    'Message Text' = (optional, any valid message text),
    'Notify Type' = ACK_NOTIFICATION,
    'To State' = (S1)
14. RECEIVE
    DESTINATION = (a device other than the TD),
    SOURCE = IUT,
    UnconfirmedEventNotification-Request,
    'Process Identifier' = (the process identifier configured for this event),
    'Initiating Device Identifier' = IUT,
    'Event Object Identifier' = (the event-initiating object),
    'Time Stamp' = (T3),
    'Notification Class' = (the notification class configured for this event),
    'Priority' = (the priority configured for this event),
    'Event Type' = (E1),
    'Notify Type' = ACK_NOTIFICATION,
    'To State' = (S1)
15. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = (TRUE, TRUE, TRUE)

~~Notes to Tester: The destination address used for the acknowledgment notification in steps 9 and 14 shall be the same address used in step 4. When multiple event notifications are expected for a specific event, the order that the IUT transmits them in is irrelevant. If the IUT can only be configured with one recipient in the Recipient_List property of the issuing Notification_class object, omit steps 4, 9, and 14.~~

9.1.2.1 Unsuccessful Alarm Acknowledgment of Confirmed Event Notifications Because the 'Time Stamp' is Too Old

Reason For Change: Made changes to include not supported transitions.

Purpose: To verify that an alarm remains unacknowledged if the time stamp in the acknowledgment does not match the most recent transition to the current alarm state.

Test Concept: An alarm is triggered that causes the IUT to notify the TD and one other device. The TD acknowledges the alarm using an old time stamp and verifies that the acknowledgment is not accepted by the IUT and that the IUT does not notify other devices that the alarm was acknowledged. The TD then acknowledges the alarm using the proper time stamp and verifies that the acknowledgment is properly noted by the IUT. The IUT notifies all other recipients that the alarm was acknowledged.

Configuration Requirements: The IUT shall be configured with at least one object that can detect alarm conditions and send confirmed notifications. The Acked_Transitions property shall have the value B'111' indicating that all transitions have been acknowledged. The TD and one other BACnet device, if the IUT supports multiple recipients, shall be recipients of the alarm notification. *D1 is either the pTimeDelay, or pTimeDelayNormal parameter, or 0 (for transitions to and from FAULT state) depending on the event transition.*

Notes to Tester: The destination address used for the acknowledgment notification in step 11 shall be the same address used in step 3. If the IUT can only be configured with one recipient in the Recipient_List property of the issuing Notification_class object, omit steps 5, 6, 15, and 16.

Test Steps:

1. MAKE (a change that triggers the detection of an alarm event in the IUT)
2. WAIT (~~pTimeDelay~~*D1*)
3. BEFORE **Notification Fail Time**
   RECEIVE ConfirmedEventNotification-Request,
   'Process Identifier' = (the process identifier configured for this event),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (the object detecting the alarm),
   'Time Stamp' = (T1: any valid time stamp),
   'Notification Class' = (the notification class configured for this event),
   'Priority' = (the priority configured for this event type),
   'Event Type' = (E1: any valid event type),
   'Message Type' = (optional, any valid message text),
   'Notify Type' = (the notify type configured for the event),
   'AckRequired' = TRUE,
   'From State' = ~~NORMAL~~*(any appropriate event state)*,
   'To State' = (S1: any appropriate ~~non-normal~~ event state),
   'Event Values' = (the values appropriate to the event type)
4. TRANSMIT BACnet-SimpleACK-PDU
5. RECEIVE
   DESTINATION = (a device other than the TD),
   SOURCE = IUT,
   ConfirmedEventNotification-Request,
   'Process Identifier' = (the process identifier configured for this event),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (the object detecting the alarm),
   'Time Stamp' = (T1),
   'Notification Class' = (the notification class configured for this event),
   'Priority' = (the priority configured for this event type),
   'Event Type' = (E2: any valid event type),
   'Message Type' = (optional, any valid message text),
   'Notify Type' = (the notify type configured for the event),
   'AckRequired' = TRUE,
   'From State' = ~~NORMAL~~*(any appropriate event state)*,
   'To State' = (S2: any appropriate ~~non-normal~~ event state),
   'Event Values' = (the values appropriate to the event type)
6. TRANSMIT BACnet-SimpleACK-PDU
7. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = ~~(FALSE,TRUE,TRUE)~~*(appropriate bit FALSE, the others TRUE)*
8. TRANSMIT AcknowledgeAlarm-Request,
   'Acknowledging Process Identifier' = (the value of the 'Process Identifier' parameter in the event notification),
   'Event Object Identifier' = (the 'Event Object Identifier' from the event notification),
   'Event State Acknowledged' = (the state specified in the 'To State' parameter of the notification),
   'Time Stamp' = (any valid time stamp),
   'Acknowledgement Source' = (any valid value),
   'Time of Acknowledgment' = (the current time using a Time format)
9. RECEIVE BACnet-Error-PDU
   Error Class = SERVICES,
   Error Code = INVALID_TIME_STAMP
10. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = (FALSE,TRUE,TRUE)
11. TRANSMIT AcknowledgeAlarm-Request,
    'Acknowledging Process Identifier' = (the process identifier configured for this event),
    'Event Object Identifier' = (the 'Event Object Identifier' from the event notification),
    'Event State Acknowledged' = (the state specified in the 'To State' parameter of the notification),
    'Time Stamp' = (the time stamp conveyed in the notification),
    'Time of Acknowledgment' = (the current time using a Time format)
12. RECEIVE BACnet-Simple-ACK-PDU
13. IF (Protocol_Revision is present and Protocol_Revision ≥ 1) THEN

BEFORE **Notification Fail Time**
RECEIVE
ConfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (T2: any valid time stamp),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event type),
'Event Type' = (any valid event type),
'Message Type' = (optional, any valid message text),
'Notify Type' = ACK_NOTIFICATION,
'To State' = (S1 or S2)

ELSE
BEFORE **Notification Fail Time**
RECEIVE
ConfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (T2: any valid time stamp),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event type),
'Event Type' = (any valid event type),
'Message Type' = (optional, any valid message text),
'Notify Type' = ACK_NOTIFICATION

14. TRANSMIT BACnet-SimpleACK-PDU
15. IF (Protocol_Revision is present and Protocol_Revision ≥ 1) THEN
RECEIVE
DESTINATION = (a device other than the TD),
SOURCE = IUT,
ConfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (T2),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event type),
'Event Type' = (any valid event type),
'Notify Type' = ACK_NOTIFICATION,
'To State' = (S1 or S2)

ELSE
RECEIVE
DESTINATION = (a device other than the TD),
SOURCE = IUT,
ConfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (T2),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event type),
'Event Type' = *(*any valid event type),
'Message Type' = (optional, any valid message text),
'Notify Type' = ACK_NOTIFICATION

16. TRANSMIT BACnet-SimpleACK-PDU
17. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = (TRUE,TRUE,TRUE)

~~Notes to Tester: The destination address used for the acknowledgment notification in step 11 shall be the same address used in step 3. If the IUT can only be configured with one recipient in the Recipient_List property of the issuing Notification_class object, omit steps 5, 6, 15, and 16.~~

9.1.2.5 Unsuccessful Alarm Acknowledgment of Unconfirmed Event Notifications Because the 'Time Stamp' is Too Old

Reason For Change: Made changes to include not supported transitions.

Purpose: To verify that an alarm remains unacknowledged if the time stamp in the acknowledgment does not match the most recent transition to the current alarm state.

Test Concept: An alarm is triggered that causes the IUT to notify the TD and one other device. The TD acknowledges the alarm using an old time stamp and verifies that the acknowledgment is not accepted by the IUT and that the IUT does not notify other devices that the alarm was acknowledged. The TD then acknowledges the alarm using the proper time stamp and verifies that the acknowledgment is properly noted by the IUT. The IUT notifies all other recipients that the alarm was acknowledged.

Configuration Requirements: The IUT shall be configured with at least one object that can detect alarm conditions and send unconfirmed notifications. The Acked_Transitions property shall have the value B'111' indicating that all transitions have been acknowledged. The TD and one other BACnet device, if the IUT supports multiple recipients, shall be recipients of the alarm notification. *D1 is either the pTimeDelay, or pTimeDelayNormal parameter, or 0 (for transitions to and from FAULT state) depending on the event transition.*

Notes to Tester: The destination address used for the acknowledgment notification in step 11 shall be the same address used in step 3. The destination address used for the acknowledgment notification in step 12 shall be the same address used in step 4. When multiple event notifications are expected for a specific event, the order that the IUT transmits them in is irrelevant. If the IUT can only be configured with one recipient in the Recipient_List property of the issuing Notification_class object, omit steps 4 and 12.

Test Steps:

1. MAKE (a change that triggers the detection of an alarm event in the IUT)
2. WAIT ~~pTimeDelay~~*D1*
3. BEFORE **Notification Fail Time**
   RECEIVE UnconfirmedEventNotification-Request,
   'Process Identifier' = (the process identifier configured for this event),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (the object detecting the alarm),
   'Time Stamp' = (T1: any valid time stamp),
   'Notification Class' = (the notification class configured for this event),
   'Priority' = (the priority configured for this event type),
   'Event Type' = (any valid event type),
   'Message Text' = (optional, any valid message text),
   'Notify Type' = (the notify type configured for the event),
   'AckRequired' = TRUE,
   'From State' = ~~NORMAL~~*(any appropriate event state)*,
   'To State' = (S1: any appropriate ~~non-normal~~ event state),
   'Event Values' = (the values appropriate to the event type)
4. IF (the notification in step 3 was not a broadcast) THEN
   RECEIVE
   DESTINATION = (at least one device other than the TD),
   SOURCE = IUT,
   UnconfirmedEventNotification-Request,
   'Process Identifier' = (the process identifier configured for this event),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (the object detecting the alarm),
   'Time Stamp' = (T1),
   'Notification Class' = (the notification class configured for this event),
   'Priority' = (the priority configured for this event type),

'Event Type' = (any valid event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = (the notify type configured for the event),
'AckRequired' = TRUE,
'From State' = ~~NORMAL~~*(any appropriate event state)*,
'To State' = (S2: any appropriate ~~non-normal~~ event state),
'Event Values' = (the values appropriate to the event type)

5. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = ~~(FALSE,TRUE,TRUE)~~*(appropriate bit FALSE, the others TRUE)*
6. TRANSMIT AcknowledgeAlarm-Request,
   'Acknowledging Process Identifier' = (the value of the 'Process Identifier' parameter in the event notification),
   'Event Object Identifier' = (the 'Event Object Identifier' from the event notification),
   'Event State Acknowledged' = (the state specified in the 'To State' parameter of the notification),
   'Time Stamp' = (a time stamp older than the one conveyed in the notification),
   'Acknowledgement Source' = (any valid value),
   'Time of Acknowledgment' = (the TD's current time using a Time format)
7. RECEIVE BACnet-Error-PDU
   Error Class = SERVICES,
   Error Code = INVALID_TIME_STAMP
8. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = (FALSE,TRUE,TRUE)
9. TRANSMIT AcknowledgeAlarm-Request,
   'Acknowledging Process Identifier' = (the process identifier configured for this event),
   'Event Object Identifier' = (the 'Event Object Identifier' from the event notification),
   'Event State Acknowledged' = (the state specified in the 'To State' parameter of the notification),
   'Time Stamp' = (the time stamp conveyed in the notification),
   'Acknowledgement Source' = (any valid value),
   'Time of Acknowledgment' = (the TD's current time using a Time format)
10. RECEIVE BACnet-Simple-ACK-PDU
11. IF (Protocol_Revision is present and Protocol_Revision ≥ 1) THEN
    BEFORE **Notification Fail Time**
    RECEIVE
    DESTINATION = LOCAL BROADCAST | GLOBAL BROADCAST | TD,
    SOURCE = IUT,
    UnconfirmedEventNotification-Request,
    'Process Identifier' = (the process identifier configured for this event),
    'Initiating Device Identifier' = IUT,
    'Event Object Identifier' = (the object detecting the alarm),
    'Time Stamp' = (T2: any valid time stamp),
    'Notification Class' = (the notification class configured for this event),
    'Priority' = (the priority configured for this event type),
    'Event Type' = (any valid event type),
    'Message Text' = (optional, any valid message text),
    'Notify Type' = ACK_NOTIFICATION,
    'To State' = (S1 or S2)

    ELSE
    BEFORE **Notification Fail Time**
    RECEIVE
    DESTINATION = LOCAL BROADCAST | GLOBAL BROADCAST | TD,
    SOURCE = IUT,
    UnconfirmedEventNotification-Request,
    'Process Identifier' = (the process identifier configured for this event),
    'Initiating Device Identifier' = IUT,
    'Event Object Identifier' = (the object detecting the alarm),
    'Time Stamp' = (T2: any valid time stamp),
    'Notification Class' = (the notification class configured for this event),
    'Priority' = (the priority configured for this event type),
    'Event Type' = (any valid event type),
    'Message Text' = (optional, any valid message text),
    'Notify Type' = ACK_NOTIFICATION
12. IF (the notification in step 11 was not broadcast) THEN

IF (Protocol_Revision is present AND Protocol_Revision ≥ 1) THEN
RECEIVE
DESTINATION = (at least one device other than the TD),
SOURCE = IUT,
UnconfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (T2),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event type),
'Event Type' = (any valid event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = ACK_NOTIFICATION,
'To State' = (S1 or S2)
ELSE
RECEIVE
DESTINATION = (at least one device other than the TD),
SOURCE = IUT,
UnconfirmedEventNotification-Request,
'Process Identifier' = (the process identifier configured for this event),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object detecting the alarm),
'Time Stamp' = (T2),
'Notification Class' = (the notification class configured for this event),
'Priority' = (the priority configured for this event type),
'Event Type' = (any valid event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = ACK_NOTIFICATION

13. VERIFY (the 'Event Object Identifier' from the event notification), Acked_Transitions = (TRUE,TRUE,TRUE)

~~Notes to Tester: The destination address used for the acknowledgment notification in step 11 shall be the same address used in step 3. The destination address used for the acknowledgment notification in step 12 shall be the same address used in step 4. When multiple event notifications are expected for a specific event, the order that the IUT transmits them in is irrelevant. If the IUT can only be configured with one recipient in the Recipient_List property of the issuing Notification_class object, omit steps 4 and 12.~~

BTL 18.1-fix1-3: Clarify How to Run the APDU Retry and Timeout Test [BTLWG-871]

Overview:
CR-0460 noted that test 13.9.1 does not work for a device that does not perform retries.

The BTLTest Plan will be updated to accommodate devices which do not perform APDU retries.

Changes:

2.1.11 Initiates Confirmed Requests

The IUT initiates any BACnet confirmed requests.

135.1-2019 - 13.9.1 - APDU Retry and Timeout		
	Test Conditionality	~~Must be executed.~~ *If the IUT cannot be configured with Number_Of_APDU_Retries greater than zero then this test shall be skipped.*
	Test Directives	
	Testing Hints	.

BTL 18.1-fix1-4: FAULT_LISTED Test Fixes [BTLWG-894, BTLWG-911, CR-0465]

Overview:
As noted in CR-0465, test 8.4.17.X9.15 requires parameters in the FAULT_OUT_OF_RANGE fault notification that the standard does not mandate.

Changes:

[Modify test 8.4.17.X9.15 in BTL Specified Tests]

8.4.17.X9.15 CHANGE_OF_RELIABILITY with the FAULT_OUT_OF_RANGE Algorithm (ConfirmedEventNotification)

Reason for Change: No test exists for this functionality.

Purpose: To verify the correct operation of the FAULT_OUT_OF_RANGE event algorithm.

Test Concept: Select a fault detecting object O1 which is configured to use the FAULT_OUT_OF_RANGE algorithm. Ensure that no other fault conditions exist in the object. Set pMonitoredValue to outside the range of values considered to be normal for the object. Verify the correct transition is generated. The fault condition is then removed. It is verified that O1 generates the correct notifications.

Test Configuration: O1 is configured to detect and report faults, to have no fault conditions present. 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.

Test Steps:
1. VERIFY pCurrentReliability = NO_FAULT_DETECTED
2. VERIFY pCurrentState = NORMAL
3. IF (pMonitoredValue is writable) THEN
   WRITE pMonitoredValue = (a value less than pMinimumNormalValue | a value greater than pMaximumNormalValue)
   ELSE
   MAKE (pMonitoredValue = a value less than pMinimumNormalValue | a value greater than pMaximumNormalValue)
4. BEFORE **Notification Fail Time**,
   RECEIVE ConfirmedEventNotification-Request,
   'Process Identifier' = (any valid process identifier),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = O1,
   'Time Stamp' = (any valid time stamp),
   'Notification Class' = (the class corresponding to the object O1 being tested),
   'Priority' = (the value configured to correspond to a TO_FAULT transition),
   'Event Type' = CHANGE_OF_RELIABILITY,
   'Message Text' = (optional, any valid message text),
   'Notify Type' = EVENT | ALARM,
   'AckRequired' = TRUE | FALSE,
   'From State' = NORMAL,
   'To State' = FAULT,
   'Event Values' = (pCurrentReliability, pStatusFlags, (A list of valid values for properties required to be reported for O1, and 0 or more other properties of O1))
5. TRANSMIT BACnet-SimpleACK-PDU
6. VERIFY pCurrentReliability = UNDER_RANGE | OVER_RANGE
7. VERIFY pCurrentState = FAULT
8. VERIFY pStatusFlags = (TRUE, TRUE,?,?)
9. IF (pMonitoredValue is writable) THEN
   WRITE pMonitoredValue = (a value greater than or equal to pMinimumNormalValue, and pMonitoredValue is less than or equal to pMaximumNormalValue)
   ELSE
   MAKE (pMonitoredValue = a value, greater than or equal to pMinimumNormalValue, and pMonitoredValue

is less than or equal to pMaximumNormalValue)

10. BEFORE **Notification Fail Time**,
    RECEIVE ConfirmedEventNotification-Request,
    'Process Identifier' = (any valid process identifier),
    'Initiating Device Identifier' = IUT,
    'Event Object Identifier' = O1,
    'Time Stamp' = (any valid time stamp),
    'Notification Class' = (the class corresponding to the object O1 being tested),
    'Priority' = (the value configured to correspond to a TO_FAULT transition),
    'Event Type' = CHANGE_OF_RELIABILITY,
    'Message Text' = (optional, any valid message text),
    'Notify Type' = EVENT | ALARM
    'AckRequired' = TRUE | FALSE,
    'From State' = FAULT,
    'To State' = NORMAL,
    'Event Values' = (pCurrentReliability, pStatusFlags, (A list of valid values for properties required to be reported for O1, and 0 or more other properties of O1))
11. TRANSMIT BACnet-SimpleACK-PDU
12. VERIFY pCurrentReliability = NO_FAULT_DETECTED
13. VERIFY pCurrentState = NORMAL
14. VERIFY pStatusFlags = (FALSE, FALSE,?, ?)

[Replace test 8.5.17.X9.15 in BTL Specified Tests]

8.5.17.X9 CHANGE_OF_RELIABILITY with the FAULT_OUT_OF_RANGE Algorithm (UnconfirmedEventNotification)

Reason for Change: No tests exists for this functionality.

Purpose: To verify the correct operation of the FAULT_OUT_OF_RANGE event algorithm.

Test Concept: Select a fault detecting object O1 which is configured to use the FAULT_OUT_OF_RANGE algorithm. Ensure that no other fault conditions exist in the object. Set pMonitoredValue to outside the range of values considered to be normal for the object. Verify the correct transition is generated. The fault condition is then removed. It is verified that O1 generates the correct notifications.

Test Configuration: O1 is configured to detect and report faults, to have no fault conditions present. 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.

Test Steps: The test steps for this test case are identical to the test steps in 8.4.17.X9.15 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 8.4.17.X9.15 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.

BTL 18.1-fix1-5: Correcting Result Flags for Logging Tests [BTLWG-969]

Overview:
CR-0246 response specified that in each of tests 7.3.2.25.1, 7.3.2.26, 7.3.2.27, 7.3.2.28, the steps need to be modified with response received in the Read Range Ack.

Changes:

[In BTL Specified Tests, modify tests 7.3.2.25.1, 7.3.2.25.2, 7.3.2.25.3, 7.3.2.225.4]

7.3.2.25.1 Internal Logging of Notifications
Reason for Change: Fixed the Result Flags value in step 10.

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

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

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

Test Steps:

1. WRITE Enable = TRUE
2. MAKE (a condition exist that will cause the device to generate an event transition)
3. WAIT D1
4. RECEIVE ConfirmedEventNotification-Request,
   'Process Identifier' = (any valid process identifier),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (any valid object),
   'Time Stamp' = (T1, any valid timestamp),
   'Notification Class' = (any valid notification class),
   'Priority' = (any valid priority),
   'Event Type' = (any standard event type),
   'Message Text' = (optional, any valid message text),
   'Notify Type' = ALARM | EVENT,
   'AckRequired' = TRUE | FALSE,
   'From State' = (state S1, any valid state for this event type),
   'To State' = (state S2, any valid state for this event type that can follow S1),
   'Event Values' = (any values appropriate to the event type)
5. TRANSMIT BACnet-SimpleACK-PDU
6. MAKE (IUT generate an EventNotification)
7. WAIT D2
8. RECEIVE ConfirmedEventNotification-Request,
   'Process Identifier' = (any valid process identifier),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' (any valid object),
   'Time Stamp' = (T2, any valid timestamp),
   'Notification Class' = (any valid notification class),
   'Priority' = (any valid priority),
   'Event Type' = (any standard event type),
   'Message Text' = (optional, any valid message text),
   'Notify Type' = ALARM | EVENT,
   'AckRequired' = TRUE | FALSE,
   'From State' = (state S3, any valid state for this event type),
   'To State' = (state S4, any valid state for this event type that can follow S3),
   'Event Values' = (any values appropriate to the event type)
9. TRANSMIT BACnet-SimpleACK-PDU
10. READ RC = LO1, Record_Count

11. TRANSMIT ReadRange-Request,
    'Object Identifier' = LO1,
    'Property Identifier' = Log_Buffer,
    'Reference Index'= RC,
    'Count' = -2
10. RECEIVE ReadRange-ACK,
    'Object Identifier' = LO1,
    'Property Identifier' = Log_Buffer,
    'Result Flags' = {~~FALSE~~?, ?, *FALSE*~~TRUE~~},
    'Item Count' = 2,
    'Item Data' = (logged data that matches the information received in steps 3 and 6, except that Process_Identifier may be any value and is not required to match)
11. CHECK (T2 > T1, and that the notifications were logged in order)

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

7.3.2.25.2 Remote Logging of Notifications
Reason for Change: Fixed Result Flags in step 8.

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

Test Concept: Make TD send multiple event notification messages. Use ReadRange to retrieve the events from an Event Log or perhaps from multiple Event Logs in the IUT, and compare the two representations.

Configuration Requirements: LO1 is an Event Log object in IUT which logs the event types which are sent. Stop_When_Full in LO1 shall be FALSE or absent.

Test Steps:

1. WRITE Enable = TRUE
2. TRANSMIT ConfirmedEventNotification-Request,
   'Process Identifier' = (any valid process identifier),
   'Initiating Device Identifier' = TD,
   'Event Object Identifier' = (any valid object identifier),
   'Time Stamp' = (T1, any valid timestamp),
   'Notification Class' = (any valid notification class),
   'Priority' = (any valid priority),
   'Event Type' = (any standard event type),
   'Message Text' = (optional, any valid message text),
   'Notify Type' = ALARM | EVENT,
   'AckRequired' = TRUE | FALSE,
   'From State' = (state S1, any valid state for this event type),
   'To State' = (state S2, any valid state for this event type that can follow S1),
   'Event Values' = (any values appropriate to the event type)
3. RECEIVE BACnet-SimpleACK-PDU
4. TRANSMIT ConfirmedEventNotification-Request,
   'Process Identifier' = (any valid process identifier),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (any valid object identifier),
   'Time Stamp' = (T2, any valid timestamp),
   'Notification Class' = (any valid notification class),
   'Priority' = (any valid priority),
   'Event Type' = (any standard event type),
   'Message Text' = (optional, any valid message text),
   'Notify Type' = ALARM | EVENT,
   'AckRequired' = TRUE | FALSE,
   'From State' = (state S3, any valid state for this event type),
   'To State' = (state S4, any valid state for this event type that can follow S3),

'Event Values' = (any values appropriate to the event type)
5. RECEIVE BACnet-SimpleACK-PDU
6. READ RC = LO1, Record_Count
7. TRANSMIT ReadRange-Request,
'Object Identifier' = LO1,
'Property Identifier' = Log_Buffer,
'Reference Index' = RC,
'Count' = -2
8. RECEIVE ReadRange-ACK,
'Object Identifier' = LO1,
'Property Identifier' = Log_Buffer,
'Result Flags' = {~~FALSE~~?, ?, *FALSE*~~TRUE~~},
'Item Count' = 2,
'Item Data' = (logged data that matches the information received in steps 2 and 4, except that Process_Identifier can be any value and is not required to match)
9. CHECK (that the events were logged in the order in which they were received)

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

7.3.2.25.3 Internal Logging of ACK_NOTIFICATIONs
Reason for Change: Removed unused step #2 and fixed the Result Flags in step 11.

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

Test Concept: Make the IUT generate an ACK_NOTIFICATION message. Use ReadRange to retrieve that same event from an Event Log and compare the two representations. If the IUT does not support logging of the ACK_NOTIFICATIONs which it initiates, this test shall be skipped.

Configuration Requirements: O1 is an event initiating object in the IUT, which is configured to send event notifications to TD. LO1 is an Event Log object in IUT which logs ACK_NOTIFICATIONs.

Test Steps:

1. WRITE Enable = TRUE
~~2. READ RC = LO1, Record_Count~~
2~~3~~. MAKE (the IUT generate a notification)
*3*4. RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (PI1, any valid process identifier),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = O1,
'Time Stamp' = (T1, any valid timestamp),
'Notification Class' = (N1, any valid notification class),
'Priority' = (P1, any valid priority),
'Event Type' = (ET1, any standard event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = ALARM | EVENT,
'AckRequired' = TRUE | FALSE,
'From State' = (S1, any valid state for this event type),
'To State' = (S2, any valid state for this event type),
'Event Values' = (any values appropriate to the event type)
4~~5~~. TRANSMIT BACnet-SimpleACK-PDU
5~~6~~. TRANSMIT AcknowledgeAlarm-Request,
'Acknowledging Process Identifier' = (any valid value),
'Event Object Identifier' = O1,
'Event State Acknowledged' = S2,
'Time Stamp' = T1,
'Time of Acknowledgment' = (the current time)
6~~7~~. RECEIVE BACnet-SimpleACK-PDU

7~~8~~. BEFORE **Notification Fail Time**
RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = PI1,
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = O1,
'Time Stamp' = (T2, any valid timestamp > T1),
'Notification Class' = N1,
'Priority' = P1,
'Event Type' = ET1,
'Message Text' = (optional, any valid message text),
'Notify Type' = ACK_NOTIFICATION,
'From State' = S1
8~~9~~. TRANSMIT BACnet-SimpleACK-PDU
9. *READ RC = LO1, Record_Count*
10. TRANSMIT ReadRange-Request,
'Object Identifier' = LO1,
'Property Identifier' = Log_Buffer,
'Reference Index' = RC,
'Count' = -1
11. RECEIVE ReadRange-ACK,
'Object Identifier' = LO1,
'Property Identifier' = Log_Buffer,
'Result Flags' = {~~FALSE~~*?*, ?, *FALSE*~~TRUE~~},
'Item Count' = 1,
'Item Data' = (logged data that matches the information received in step 7~~4~~,
except that Process_Identifier can be any value and is not
required to match)

Notes to Tester: When the UnconfirmedEventNotification service is used instead of the ConfirmedEventNotification service, the TD shall skip the steps in which *BACnet*-SimpleACK-PDUs are sent in response to ConfirmedEventNotifications.

7.3.2.25.4 Remote Logging of ACK_NOTIFICATIONs
Reason for Change: Fixed Result Flags in step 6.

Purpose: To verify that the IUT correctly collects and represents ACK_NOTIFICATIONs which it receives.

Test Concept: Send an ACK_NOTIFICATION to the IUT. Use ReadRange to retrieve that same event from an Event Log, and compare the two representations.

Configuration Requirements: LO1 is an Event Log object in IUT which logs ACK_NOTIFICATIONs. Stop_When_Full in LO1 shall be FALSE or absent.

Test Steps:

1. WRITE Enable = TRUE
2. TRANSMIT ConfirmedEventNotification-Request,
'Process Identifier' = (any valid process identifier),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (any valid object identifier),
'Time Stamp' = (T1, any valid timestamp),
'Notification Class' = (any valid notification class),
'Priority' = (any valid priority),
'Event Type' = (any standard event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = ACK_NOTIFICATION,
'From State' = (state S1, any valid state for this event type)
3. RECEIVE BACnet-SimpleACK-PDU
4. READ RC = LO1, Record_Count
5. TRANSMIT ReadRange-Request,

'Object Identifier' = LO1,
'Property Identifier' = Log_Buffer,
'Reference Index' = RC,
'Count' = -1

6. RECEIVE ReadRange-ACK,
'Object Identifier' = LO1,
'Property Identifier' = Log_Buffer,
'Result Flags' = {~~FALSE~~?, ?, *FALSE*~~TRUE~~},
'Item Count' = 1,
'Item Data' = (logged data that matches the information received in step 2,
except that Process_Identifier can be any value and is not required to match)

Notes to Tester: When the UnconfirmedEventNotification service is used instead of the ConfirmedEventNotification service, the test shall skip the step in which a *BACnet*-SimpleACK-PDU is expected.

BTL 18.1-fix1-6: Foreign Device Registration [BTLWG-1001]

Overview:
The Network Port test addition reference but do not define Foreign Device Registration Fail Time.

This proposal adds the definition for this new timer into the BTL Specified Tests.

Changes:

[In BTL Specified Tests, modify Clause 4.5.9]

[Add the following into the list of timers]

Foreign Device Registration Fail Time: ❑↵

[Add Clause 6.3.X4 into BTL Specified Tests]

6.3.X4 **Foreign Device Registration Fail Time**

The **Foreign Device Registration Fail Time** is the elapsed time, in seconds, between when the device sends a Register-Foreign-Device request and when the device considers the request to have failed due to having not received a BVLC-Result for the registration request.

BTL 18.1-fix1-7: BBMDs are Required to Persist BDTs [BTLWG-1009]

Overview:
In the response to CR-0478, test 14.3.3 is incorrect. BBMD's claiming Protocol Revision 16 and prior are expected to accept and persist values via the Write-Broadcast-Distribution-Table service. This means that the test should not allow for IUTs, claiming PR16 or less, to not execute Write-BDT.

Changes:
[In BTL Specified Tests 16.1, 14.3.3 Verify Broadcast Distribution Table Created from the Configuration Saved During the Previous Session, Page No. 380]

14.3.3 Verify Broadcast Distribution Table Created from the Configuration Saved During the Previous Session
Reason for Change: ~~Revised test to allow testing when BDT can be configured with local configuration tool only.~~ *IUTs claiming PR16 and less need to accept Write-Broadcast-Distribution-Table message.*

Purpose: To verify that a BBMD will update the BDT in the local configuration database and initialize it at startup.

Configuration Requirements: The IUT's BDT does not consist of the same entries *that are* ~~as are can either be~~ written in step 1~~, or configured with a local configuration tool~~.

Test Steps:

1. ~~IF (The IUT's BDT can be written with Write-Broadcast-Distribution-Table) THEN~~
——TRANSMIT
DA = IUT,
SA = D1,
Write-Broadcast-Distribution-Table,
(List of BDT entries at least one of which is different from what it has
IUT 255.255.255.255
BBMD1 255.255.255.255
BBMD2 255.255.255.255
)
~~2.~~ RECEIVE
DA = D1,
SA = IUT,
BVLC-Result,
'Result Code' = Successful completion
~~ELSE~~
~~2. MAKE (the IUT's BDT different, so that values in the BDT at step 6 can be distinguished)~~
2. WAIT (Vendor specified period for BDT to be saved in non-volatile memory)
3. MAKE (the IUT reset)
4. TRANSMIT
DA = IUT,
SA = D1,
Read-Broadcast-Distribution-Table
5. RECEIVE
DA = D1,
SA = IUT,
Read-Broadcast-Distribution-Table-Ack,
List of BDT Entries
6. CHECK (*IUT's BDT holds the entries with which it was configured in step 1)*~~List of BDT Entries consisting of three entries (order unspecified)~~

~~IUT~~	~~255.255.255.255~~
~~BBMD1~~	~~255.255.255.255~~
~~BBMD2~~	~~255.255.255.255~~

~~)~~