BACnet® TESTING LABORATORIES ADDENDA

Addendum fix3 to BTL Test Package 23.3

Revision v4
Revised 7/25/2024

Approved by the BTL Working Group on July 25, 2024
Approved by the BTL Working Group Voting Members on August 28, 2024;
Published on August 30, 2024.

[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-23.3 fix3-1: Clarify OFFNORMAL vs offnormal Usage in Test 8.5.17.10 [BTLWG-1314].....2
BTL-23.3 fix3-2: Fix Test 7.3.2.30.6 Test Setup [BTLWG-1315].....5
BTL-23.3 fix3-3: Fix Test Plan Reference and CHECK Step to Reference in 7.3.2.40.1.6 Tests [BTLWG-1482].....7
BTL-23.3 fix3-4: Fix List_Of_Object_Property_Reference Internal Test to match Test Concept [BTLWG-1485].....10
BTL-23.3 fix3-5: Fix Router Reply Postponed Test [BTLWG-1497].....12
BTL-23.3 fix3-6: ACTIVATE_CHANGES is required if NPO is Configurable [BTLWG-1547].....13
BTL-23.3 fix3-7: Must Understand for Unknown Data Options [BTLWG-1548].....17
BTL-23.3 fix3-8: Must Understand for Unknown data Options - Router [BTLWG-1549].....19
BTL-23.3 fix3-9: Must Understand for Unknown Data Options - Hub [BTLWG-1561].....22

In the following document, language to be added to existing clauses within the BTL Test Package 23.3 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-23.3 fix3-1: Clarify OFFNORMAL vs offnormal Usage in Test 8.5.17.10 [BTLWG-1314]

Overview:

Test 8.5.17.10 from "BTL Specified Tests-20.0.1_final.pdf" has in Test Concept:
... Make O1 transition to an OFFNORMAL state ...
... verify that the object transitions from NORMAL to the original OFFNORMAL state.

A couple of problems in the test steps:

- After steps 11 and 12 it continues with 11 and twelve; should be 13 and 14.
- 'To State' = OFFNORMAL in Steps 4 and 12 (1st incarnation) as well as VERIFY pCurrentState = OFFNORMAL in steps 5 and 12 (2nd incarnation) should be worded to accept any offnormal, not only specific OFFNORMAL.
- According to test concept the same offnormal state is expected in all steps. Currently the test steps contain nothing relating to this part of the concept.

The last problem is the serious one; the other could be considered errata.

To fix this the 4 steps could all use a symbolic name “S1” instead of "OFFNORMAL" S1 would be the specific offnormal for that test. S1 could be defined already in MAKE step 3 as a configuration option for the test or it could be identified in the first occurrence in step 4.

Changes:

Checklist Changes

None

Test Plan Changes

[Modify all usages of 135.1-2023 - 8.5.17.10 to BTL - 8.5.17.10. There are currently 2 of them.]

Specified Test Changes

[Copy test 8.5.17.10 from 135.1-2023 and modify test as shown]

8.5.17.10 After FAULT-to-NORMAL, Re-Notification of OFFNORMAL (UnconfirmedEventNotifications)
Reason for Change: *Clarifying that the same offnormal state should be used throughout the test.*

Purpose: To verify that objects go to the NORMAL state after leaving the FAULT state, then transition to OFFNORMAL if the condition still exists.

Test Concept: Select a fault detecting object O1 which is able to detect ~~OFFNORMAL~~offnormal conditions. Make O1 transition to an ~~OFFNORMAL~~*offnormal* state (*S1)* and then transition to FAULT. Remove the condition causing the FAULT and verify O1 transitions from FAULT to NORMAL, then verify that the object transitions from NORMAL to the original ~~OFFNORMAL~~*offnormal* state.

Configuration Requirements: O1 is configured to detect and report unconfirmed events and faults. O1 is configured to have no fault conditions present, and Event_State is NORMAL. The 'Issue Confirmed Notifications' parameter shall have a value of FALSE.

Test Steps:
1. VERIFY pCurrentReliability = NO_FAULT_DETECTED
2. VERIFY pCurrentState = NORMAL
3. MAKE(O1 transition to ~~an offnormal state~~*S1*)

4. WAIT pTimeDelay
5. BEFORE **Notification Fail Time**
   RECEIVE UnconfirmedEventNotification-Request
   'Process Identifier' = (any valid process identifier),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = O1,
   'Time Stamp' = (any valid time stamp),
   'Notification Class' = (the notification class configured for O1),
   'Priority' = (the value configured for the transition),
   'Event Type' = (ET1, any valid off normal event type),
   'Message Text' = (optional, any valid message text),
   'Notify Type' = ALARM | EVENT,
   'AckRequired' = TRUE | FALSE,
   'From State' = NORMAL,
   'To State' = ~~OFFNORMAL~~*S1*,
   'Event Values' = (property-values appropriate for O1)
6. VERIFY pCurrentState = ~~OFFNORMAL~~*S1*
7. MAKE(O1 enter a fault state)
8. BEFORE **Notification Fail Time**
   RECEIVE UnconfirmedEventNotification-Request
   'Process Identifier' = (any valid process identifier),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = O1,
   'Time Stamp' = (any valid timestamp),
   'Notification Class' = (the notification class configured for O1),
   'Priority' = (the value configured for the transition),
   'Event Type' = CHANGE_OF_RELIABILITY,
   'Message Text' = (optional, any valid message text),
   'Notify Type' = ALARM | EVENT,
   'AckRequired' = TRUE | FALSE,
   'From State' = ~~OFFNORMAL~~*S1*,
   'To State' = FAULT,
   'Event Values' = ((R1 any valid BACnetReliability *other than NO_FAULT_DETECTED*),
   (~~?~~*T*, T, ?, ?),
   (A list of valid values for properties required to be reported for O1,
   and 0 or more other properties of O1))
9. MAKE(O1 clear the fault condition)
10. 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 time stamp),
    'Notification Class' = (the notification class configured for O1),
    'Priority' = (the value configured for the transition),
    'Event Type' = CHANGE_OF_RELIABILITY,
    'Message Text' = (optional, any valid message text),
    'Notify Type' = ALARM | EVENT,
    'AckRequired' = TRUE | FALSE,
    'From State' = FAULT,
    'To State' = NORMAL,
    'Event Values' = (NO_FAULT_DETECTED,
    (F, F, ?, ?),
    (A list of valid values for properties required to be reported for O1,
    and 0 or more other properties of O1))

~~*11. VERIFY pCurrentReliability = NO_FAULT_DETECTED*~~

~~12. VERIFY pCurrentState = NORMAL~~

11.~~13.~~ WAIT until TS1 + pTimeDelay

12. BEFORE **Notification Fail Time**

RECEIVE UnconfirmedEventNotification-Request
'Process Identifier' = (any valid process identifier),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = O1,
'Time Stamp' = (any valid time stamp),
'Notification Class' = (the notification class configured for O1),
'Priority' = (the value configured for the transition),
'Event Type' = ET1,
'Message Text' = (optional, any valid message text),
'Notify Type' = ALARM | EVENT,
'AckRequired' = TRUE | FALSE,
'From State' = NORMAL,
'To State' = ~~OFFNORMAL~~*S1*,
'Event Values' = (property-values appropriate for O1)

~~15.~~*13*. VERIFY pCurrentReliability = NO_FAULT_DETECTED

~~16.~~*14*. VERIFY pCurrentState = ~~OFFNORMAL~~*S1*

BTL-23.3 fix3-2: Fix Test 7.3.2.30.6 Test Setup [BTLWG-1315]

Overview:

This test uses Configuration Base Setup 2 for notification testing. Base Setup 2 defines a DS (Device notification source) Device, an IUT (instrument under test) Device, and a single destination Device D1. The overview should only refer to the devices defined in the base setup - D1 as the destination device. Any reference to Device D2 should be removed, as D2 is not defined or used according to Base Setup 2. Using only the devices defined in the base setup will avoid potential confusion.

Changes:

Checklist Changes

None

Test Plan Changes

[In Test Plan 3.51.6 change 135.1 2023 - 7.3.2.30.6 to BTL - 7.3.2.30.6]

Specified Test Changes

[Copy test 7.3.2.30.6 from 135.1-2023 and then modify Test Steps as shown]

7.3.2.30.6 Out_of_Service Property Test

Reason for Change: Corrected inconsistencies in test steps with Base Setup 2.

Purpose: This test case verifies that event forwarding is not done while Out_Of_Service is TRUE.

Test Concept: Set up both Recipient_List and Subscribed_Recipient recipient entries with no filters specified and then send event notifications to the Notification Forwarder while the value of the Out_Of_Service property is TRUE. Subscribed_Recipients are configured as part of base setup 2 for Notification Forwarder object tests. Verify that forwarding of the event notifications is not performed.

Configuration Requirements: The selected object is configured such that its Out_Of_Service shall be set to FALSE and Reliability set to NO_FAULT_DETECTED. Base setup 2 for Notification Forwarder object tests with TR lifetime sufficient for this test.

Test Steps:

1. MAKE (Recipient_List = {(all), -- Valid Days
(all), -- From Time, To Time
DEST_OBJ_ID~~2~~, -- Recipient ~~D2~~*D1*
DEST_PROCESS_ID, -- Process Identifier
FALSE, -- Issue Confirmed Notifications
{T, T, T} -- Transitions
}) -- One list element
2. MAKE (Out_Of_Service = TRUE)
3. VERIFY Out_Of_Service = TRUE
4. VERIFY Status_Flags = (FALSE, FALSE, FALSE, TRUE)
5. TRANSMIT SOURCE = DS, UnconfirmedEventNotification-Request,
'Process Identifier' = SRC_PROCESS_ID,
'Initiating Device Identifier' = SRC_NOTIF_DEV,
'Event Object Identifier' = SRC_NOTIF_OBJ,
'Time Stamp' = (any valid time stamp),
'Notification Class' = SRC_NOTIF_CLS,
'Priority' = (any valid priority),
'Event Type' = (any valid event type),
'Message Text' = (optional, any valid message text),

'Notify Type' = SRC_NOTIF_TYP,
'AckRequired' = (any valid value), -- absent if Notify Type is ACK_NOTIFICATION
'From State' = (any valid From_State), -- absent if Notify Type is ACK_NOTIFICATION
'To State' = (any valid To_State),
'Event Values' = (any valid event values) -- absent if Notify Type is ACK_NOTIFICATION

6. WAIT Notification Fail Time
7. CHECK (the IUT did not transmit an event notification)
8. MAKE (Out_Of_Service = FALSE)
9. VERIFY Out_Of_Service = FALSE
10. VERIFY Status_Flags = (?, ?, ?, FALSE)
11. TRANSMIT SOURCE = DS, UnconfirmedEventNotification-Request,
    'Process Identifier' = SRC_PROCESS_ID,
    'Initiating Device Identifier' = SRC_NOTIF_DEV,
    'Event Object Identifier' = SRC_NOTIF_OBJ,
    'Time Stamp' = (any valid time stamp),
    'Notification Class' = SRC_NOTIF_CLS,
    'Priority' = (any valid priority),
    'Event Type' = (any valid event type),
    'Message Text' = (optional, any valid message text),
    'Notify Type' = SRC_NOTIF_TYP,
    'AckRequired' = (any valid value), -- absent if Notify Type is ACK_NOTIFICATION
    'From State' = (any valid From_State), -- absent if Notify Type is ACK_NOTIFICATION
    'To State' = (any valid To_State),
    'Event Values' = (any valid event values) -- absent if Notify Type is ACK_NOTIFICATION
12. BEFORE **Notification Fail Time** ~~-- The following can be in any order~~
    RECEIVE DESTINATION = D1, UnconfirmedEventNotification-Request
    ~~RECEIVE DESTINATION = D2, UnconfirmedEventNotification-Request~~
13. MAKE (Out_Of_Service = TRUE)
14. VERIFY Out_Of_Service = TRUE
15. VERIFY Status_Flags = (FALSE, FALSE, FALSE, TRUE)
16. IF (Reliability is writable) THEN
    REPEAT X = (all values of the Reliability enumeration appropriate to the object type except
    NO_FAULT_DETECTED) DO {
    WRITE Reliability = X
    VERIFY Reliability = X
    VERIFY Status_Flags =(?, TRUE, ?, TRUE)
    WRITE Reliability = NO_FAULT_DETECTED
    VERIFY Reliability = NO_FAULT_DETECTED
    VERIFY Status_Flags = (?, FALSE, ?, TRUE)
    }
17. IF (Out_Of_Service is writable) THEN
    WRITE Out_Of_Service = FALSE
    ELSE
    MAKE (Out_Of_Service = FALSE)
18. VERIFY Out_Of_Service = FALSE
19. VERIFY Status_Flags = (?, FALSE, ?, FALSE)

BTL-23.3 fix3-3: Fix Test Plan Reference and CHECK Step to Reference in 7.3.2.40.1.6 Tests [BTLWG-1482]

Overview:

Affected is Test 135.1-2023 - 7.3.2.40.1.6 - Door_Unlock_Delay_Time Test in chapter 3.42.7 Supports Door_Unlock_Delay_Time Property.
Step 22 is: "CHECK (that the door control output is in a state that causes the door to be locked)".
In this case, "locked" should be replaced with "unlocked".

There is still a mistake in the test plan.
The name in the BTL testplan is :
135.1-2023 - 7.3.2.40.1.6 - Door_Unlock_Delay_Time Test
and in ASHRAE 135.1-2023 :
7.3.2.40.6 Door_Unlock_Delay_Time Test

Same mistake in
135.1-2023 - 7.3.2.40.1.1 - Commandable Present_Value Test
135.1-2023 - 7.3.2.40.1.2 - Door_Status, Lock_Status and Door_Alarm_State Tests
135.1-2023 - 7.3.2.40.1.3 - Door_Status with Physical Door Status Tests
135.1-2023 - 7.3.2.40.1.4 - Lock_Status Tests
135.1-2023 - 7.3.2.40.1.5 - Secured_Status Tests
135.1-2023 - 7.3.2.40.1.7 - Masked_Alarm_Values Test
135.1-2023 - 7.3.2.40.1.8 - Door_Open_Too_Long Test

Changes:

Checklist Changes

None

Test Plan Changes

3.42.1 Base Requirements

Base requirements must be met by any IUT that supports Access Door objects

135.1-2023 - 7.3.2.40.~~1.~~1 - Commandable Present_Value Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

…

3.42.3 Supports Configurable Out_Of_Service Property

The Out_Of_Service property in Access Door objects contained in the IUT are either writable or can be modified by any other means.

…		
135.1-2023 - 7.3.2.40.~~1.~~2 - Door_Status, Lock_Status and Door_Alarm_State Tests		
	Test Conditionality	If neither Door_Status, Lock_Status nor Door_Alarm_State is supported, this test shall be skipped.
	Test Directives	
	Testing Hints	

3.42.4 Supports Door_Status Property

The IUT contains or can be made to contain Door_Status property which is writable when Out_Of_Service is True.

135.1-2023 - 7.3.2.40.~~1.~~3 - Door_Status with Physical Door Status Tests		
	Test Conditionality	If the Door_Status property is permanently configured to have the value UNUSED then this test shall be skipped.
	Test Directives	
	Testing Hints	

3.42.5 Supports Lock_Status Property

The IUT contains or can be made to contain Lock_Status property which is writable when Out_Of_Service is True.

135.1-2023 - 7.3.2.40.~~1.~~4 - Lock_Status Tests		
	Test Conditionality	If the physical lock cannot be manipulated without writing to Present_Value of the associated Access Door object then this test shall be skipped.
	Test Directives	
	Testing Hints	

3.42.6 Supports Secured_Status Property

The IUT contains or can be made to contain Secured_Status property.

135.1-2023 - 7.3.2.40.~~1.~~5 - Secured_Status Tests		
	Test Conditionality	If the Secured_Status property is permanently configured to have the value UNKNOWN then this test shall be omitted.
	Test Directives	
	Testing Hints	

3.42.7 Supports Door_Unlock_Delay_Time Property

The IUT contains or can be made to contain a writable or read-only Door_Unlock_Delay_Time property.

~~135.1-2023~~BTL - 7.3.2.40.~~1.6~~ - Door_Unlock_Delay_Time Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

3.42.8 Supports Masked_Alarm_Values Property

The IUT contains or can be made to contain Masked_Alarm_Value property.

135.1-2023 - 7.3.2.40.~~1.~~7 - Masked_Alarm_Values Test		
	Test Conditionality	If Out_Of_Service is not writeable and cannot be set to TRUE by any other means, this test shall be skipped.
	Test Directives	
	Testing Hints	

3.42.9 Supports Intrinsic Reporting

The IUT supports intrinsic reporting.

135.1-2023 - 7.3.2.40.~~1.~~8 - Door_Open_Too_Long Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

Specified Test Changes

[Add test 7.3.2.40.6 from 135.1-2023 into BTL Specified Tests and change as shown.]

7.3.2.40.6 Door_Unlock_Delay_Time Test
Reason for Change: Step 22 required the door to be in the incorrect state.

Purpose: To verify that when the Door_Unlock_Delay_Time property has a non-zero value, the output is delayed in unlocking when a PULSE_UNLOCK or EXTENDED_PULSE_UNLOCK is written to the Present_Value and not when UNLOCK is written.

Test Concept: When unlocking the door by writing PULSE_UNLOCK to the Present_Value of the Access Door object, it is verified that the door is still locked for the specified Door_Pulse_Time then the door is unlocked. The same test is done for EXTENDED_PULSE_UNLOCK, but this time it is verified that the door is still locked for the specified Door_Extended_Pulse_Time then the door is unlocked.

Configuration Requirements: The IUT shall be configured with a door control output that can be observed during the test. The Relinquish_Default shall have the value LOCK. All writes to the Present_Value shall be performed at a priority higher than any internal algorithms writing to this property. Door_Unlock_Delay_Time shall be set to a non-zero value which is sufficient to observe the delay and check the status of the lock. Out_Of_Service shall be set to FALSE. Prior to the test the Present_Value shall have the value LOCK and the IUT is in a state that would cause the door to be locked.

Test Steps:
-- Test PULSE_UNLOCK
1. WRITE Present_Value = PULSE_UNLOCK
2. WAIT (Internal Processing Fail Time)
3. BEFORE Door_Unlock_Delay_Time
    IF (Lock_Status is present) THEN
        VERIFY Lock_Status = LOCKED
    CHECK (that the door control output is in a state that would cause the door to be locked)

4. IF (Lock_Status is present) THEN
    VERIFY Lock_Status = UNLOCKED
5. CHECK (that the door control output is in a state that would cause the door to be unlocked)
6. WAIT (Door_Pulse_Time)
7. VERIFY Present_Value = LOCK
8. IF (Lock_Status is present) THEN
    VERIFY Lock_Status = LOCKED
9. CHECK (that the door control output is in a state that would cause the door to be locked)

-- Test EXTENDED_PULSE_UNLOCK
10. WRITE Present_Value = EXTENDED_PULSE_UNLOCK
11. WAIT (Internal Processing Fail Time)
12. BEFORE Door_Unlock_Delay_Time
    IF (Lock_Status is present) THEN
        VERIFY Lock_Status = LOCKED
    CHECK (that the door control output is in a state that would cause the door to be locked)

13. IF (Lock_Status is present) THEN
    VERIFY Lock_Status = UNLOCKED
14. CHECK (that the door control output is in a state that would cause the door to be unlocked)
15. WAIT (Door_Extended_Pulse_Time)
16. VERIFY Present_Value = LOCK
17. IF (Lock_Status is present) THEN
    VERIFY Lock_Status = LOCKED
18. CHECK (that the door control output is in a state that would cause the door to be locked)

-- Test UNLOCK
19. WRITE Present_Value = UNLOCK
20. WAIT (Internal Processing Fail Time)
21. IF (Lock_Status is present) THEN
    VERIFY Lock_Status = UNLOCKED
22. CHECK (that the door control output is in a state that would cause the door to be ~~locked~~ *unlocked*)

BTL-23.3 fix3-4: Fix List_Of_Object_Property_Reference Internal Test to match Test Concept [BTLWG-1485]

Overview:

Both tests use incorrect notation in Step 1
The test concept specifies: The clock of the test object is then set to a time between two planned write operations.
The configuration requirements specify that a time Dt is defined that lies between D1 and D2.
This means that Dt must be queried in step 1.

Changes:

Checklist Changes

None

Test Plan Changes

[Change test reference for 135.1-2023 - 7.3.2.23.7 and 135.1-2023 - 7.3.2.23.8 to BTL.]

6.4.6 Supports Non-Empty List_Of_Object_Property_References Property

BTL - 7.3.2.23.7 - List_Of_Object_Property_Reference Internal Test		
	Test Conditionality	This test shall be executed if and only if the IUT is prior to protocol revision 4. If the IUT is of the correct Protocol_Revision, the IUT is required to be configurable such that this test can be run. This test may not be skipped.
	Test Directives	
	Testing Hints	

6.5.1 Base Requirements

BTL - 7.3.2.23.8 - List_Of_Object_Property_Reference External Test		
	Test Conditionality	This test shall be executed if and only if the IUT is prior to protocol revision 4. If the IUT is of the correct Protocol_Revision, the IUT is required to be configurable such that this test can be run. This test may not be skipped.
	Test Directives	
	Testing Hints	

Specified Test Changes

[Copy test 7.3.2.23.7 and 7.3.2.23.8 from 135.1-2023 into BTL Test Package and modify as shown.]

7.3.2.23.7 List_Of_Object_Property_Reference Internal Test

Reason for Change: Test changed to match the test concept.

Purpose: To verify that the Schedule object writes to objects and properties contained within the IUT.

Test Concept: The Schedule object is configured to write to a property of another object within the same device. The IUT's clock is then set to a time between a pair of scheduled write operations, and verification of the first write operation's data value is performed. The time is advanced to the second time, the Schedule object's Present_Value is checked, and verifications of the write operations are performed. If the IUT does not support writing to object properties within the IUT, then this test shall not be performed.

Configuration Requirements: The IUT is configured with a Schedule object containing a List_Of_Object_Property_ References property that references, if possible, at least one property in another object within the IUT. The Schedule object is configured with either a Weekly_Schedule or an active Exception_Schedule, during a period where Effective_Period is active, with at least two consecutive entries with distinguishable values in the List of BACnetTimeValues, and with no Exception_Schedules at a higher priority. D_1 represents the date and time of the first of these two BACnetTimeValues, with corresponding value V_1, while D_2 and V_2 (a value distinguishable from V_1) represent the second BACnetTimeValue. A time D_t is defined to occur between D_1 and D_2.

Test Steps:

1. (TRANSMIT TimeSynchronization-Request, 'Time' = D_t) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = ~~D_1~~ D_t) |
   MAKE (the local date and time = D_t)
2. WAIT Schedule Evaluation Fail Time
3. VERIFY Present_Value = V_1
4. VERIFY (value of referenced property in IUT) = V_1
5. (TRANSMIT TimeSynchronization-Request, 'Time' = D_2) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_2) |
   MAKE (the local date and time = D_2)
6. WAIT Schedule Evaluation Fail Time
7. VERIFY Present_Value = V_2
8. VERIFY (value of referenced property in IUT) = V_2

7.3.2.23.8 List_Of_Object_Property_Reference External Test

Reason for Change: Test changed to match the test concept.

Purpose: To verify that the Schedule object writes to objects and properties contained within the IUT.

Test Concept: The Schedule object is configured to write to a property of another object within the same device. The IUT's clock is then set to a time between a pair of scheduled write operations, and verification of the first write operation's data value is performed. The time is advanced to the second time, the Schedule object's Present_Value is checked, and verifications of the write operations are performed. If the IUT does not support writing to object properties within the IUT, then this test shall not be performed.

Configuration Requirements: The IUT is configured with a Schedule object containing a List_Of_Object_Property_ References property that references, if possible, at least one property in another object within the IUT. The Schedule object is configured with either a Weekly_Schedule or an active Exception_Schedule, during a period where Effective_Period is active, with at least two consecutive entries with distinguishable values in the List of BACnetTimeValues, and with no Exception_Schedules at a higher priority. D_1 represents the date and time of the first of these two BACnetTimeValues, with corresponding value V_1, while D_2 and V_2 (a value distinguishable from V_1) represent the second BACnetTimeValue. A time D_t is defined to occur between D_1 and D_2.

Test Steps:

1. (TRANSMIT TimeSynchronization-Request, 'Time' = ~~D_1~~ D_t) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = ~~D_1~~ D_t) |
   MAKE (the local date and time = ~~D_1~~ D_t)
2. WAIT Schedule Evaluation Fail Time
3. VERIFY Present_Value = V_1
4. VERIFY (value of referenced property in IUT) = V_1
5. (TRANSMIT TimeSynchronization-Request, 'Time' = D_2) |
   (TRANSMIT UTCTimeSynchronization-Request, 'Time' = D_2) |
   MAKE (the local date and time = D_2)
6. WAIT Schedule Evaluation Fail Time
7. VERIFY Present_Value = V_2
8. VERIFY (value of referenced property in IUT) = V_2

BTL-23.3 fix3-5: Fix Router Reply Postponed Test [BTLWG-1497]

Overview:
BTL Specified Tests-23.1_final: Test 12.1.X MS/TP Router Reply Postponed Test:

Step 3 involves a "Hop Count". As per the standard, "The Hop Count field shall be present only if the message is destined for a remote network, i.e. if DNET is present."
In this case, since the BACnet-Confirmed-Request is already in the destination network at Step 3, there is no need for a "Hop Count," and therefore, it should not be included in the message.

Changes:

Checklist Changes

None

Test Plan Changes

None

Specified Test Changes

[In BTL Specified Tests, modify test as shown]

12.1.X MS/TP Router Reply Postponed Test.
Reason for Change: There is no test for this functionality. Checking the hop count is unnecessary in step 4

Purpose: To verify that the IUT sends 'reply postponed' when it receives a MS/TP packet with a data_expecting_reply parameter set to TRUE destined for another network.

Test Concept: The IUT is configured to route between Network 1 on port A (MS/TP network) and Network 2 on port B (any BACnet data link). D1A resides on Network 1 and D2C resides on Network 2. The IUT receives a message from D1A destined for D2C with the data expecting reply parameter set to TRUE. The IUT transmits ‘Reply Postponed’ to D1A before attempting to route the message.

Configuration Requirements: The IUT is actively routing between Network 1 and Network 2 and D1A has discovered D2C.

Test Steps:

1. TRANSMIT PORT A
   SA = D1A
   DA = IUT
   DNET = Network 2
   DADR = D2C
   Hop Count = 255
   BACnet-Confirmed-Request-PDU
2. RECEIVE PORT A
   Reply Postponed
3. RECEIVE PORT B
   SA = IUT
   DA = D2C
   SNET = Network 1
   SADR = D1A
   ~~Hop Count = (any integer x: 0 < x < 255)~~
   BACnet-Confirmed-Request-PDU

BTL-23.3 fix3-6: ACTIVATE_CHANGES is required if NPO is Configurable [BTLWG-1547]

Overview:

K.5.16 BIBB - Device Management-ReinitializeDevice-B (DM-RD-B) states, " If the device supports a configurable Network Port Object, then it shall also support the restart choice ACTIVATE_CHANGES."

To be fully interoperable with newer and older clients, the IUT must support receiving either WARMSTART or ACTIVATE_CHANGES.

Changes:

Checklist Changes

None

Test Plan Changes

9.1 Data Link Layer - MS/TP - Master Node

9.1.8 Supports Configuration Through Network Port Object

The IUT supports full, or partial, configuration of the data link through the Network Port object.

…		
135.1-2023 - 7.3.2.46.1.1 - Configure Network Through Network Port Object Test		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision 17 or higher.
	Test Directives	Execute this test at least once on each Network Port object that has Network_Type = MSTP and contains writable properties. Execute test 7.3.2.46.1.1 at least once with ReinitializeDevice WARMSTART and at least once with ACTIVATE_CHANGES.
	Testing Hints	

9.2 Data Link Layer - MS/TP - Slave Node

9.2.2 Supports Configuration Through Network Port Object

The IUT supports full, or partial, configuration of the data link through the Network Port object. Specifically, at least 1 property in the Network Port object which changes the behavior of the data link is writable.

135.1-2023 - 7.3.2.46.1.1 - Configure Network Through Network Port Object Test		
	Test Conditionality	Must be executed.
	Test Directives	Perform at least once. Repeat each time the network is reconfigured for a test. Execute test 7.3.2.46.1.1 at least once with ReinitializeDevice WARMSTART and at least once with ACTIVATE_CHANGES.
	Testing Hints	

9.3 Data Link Layer - IPv4

9.3.5 Supports Configuration Through Network Port Object

The IUT supports full, or partial, configuration of the data link through the Network Port object.

…		
135.1-2023 - 7.3.2.46.1.1 - Configure Network Through Network Port Object Test		
	Test Conditionality	Must be executed if Protocol_Revision is 17 or higher..
	Test Directives	Execute this test at least once on each Network Port object that has Network_Type = IPV4 and contains writable properties. Execute test 7.3.2.46.1.1 at least once with ReinitializeDevice WARMSTART and at least once with ACTIVATE_CHANGES.
	Testing Hints	

9.4 Data Link Layer - ZigBee

9.4.2 Supports Configuration Through Network Port Object

The IUT supports full, or partial, configuration of the data link through the Network Port object. Specifically, at least 1 property in the Network Port object which changes the behavior of the data link is writable.

135.1-2023 - 7.3.2.46.1.1 - Configure Network Through Network Port Object Test		
	Test Conditionality	Must be executed.
	Test Directives	Perform at least once. Repeat each time the network is reconfigured for a test. Execute test 7.3.2.46.1.1 at least once with ReinitializeDevice WARMSTART and at least once with ACTIVATE_CHANGES.
	Testing Hints	

9.5 Data Link Layer - Ethernet

9.5.2 Supports Configuration Through Network Port Object

The IUT supports full, or partial, configuration of the data link through the Network Port object. Specifically, at least 1 property in the Network Port object which changes the behavior of the data link is writable.

135.1-2023 - 7.3.2.46.1.1 - Configure Network Through Network Port Object Test		
	Test Conditionality	Must be executed.
	Test Directives	Perform at least once. Repeat each time the network is reconfigured for a test. Execute test 7.3.2.46.1.1 at least once with ReinitializeDevice WARMSTART and at least once with ACTIVATE_CHANGES.
	Testing Hints	

9.6 Data Link Layer - ARCNET

9.6.2 Supports Configuration Through Network Port Object

The IUT supports full, or partial, configuration of the data link through the Network Port object. Specifically, at least 1 property in the Network Port object which changes the behavior of the data link is writable.

135.1-2023 - 7.3.2.46.1.1 - Configure Network Through Network Port Object Test		
	Test Conditionality	Must be executed.
	Test Directives	Perform at least once. Repeat each time the network is reconfigured for a test. Execute test 7.3.2.46.1.1 at least once with ReinitializeDevice WARMSTART and at least once with ACTIVATE_CHANGES.
	Testing Hints	

9.7 Data Link Layer - LonTalk

9.7.2 Supports Configuration Through Network Port Object

The IUT supports full, or partial, configuration of the data link through the Network Port object. Specifically, at least 1 property in the Network Port object which changes the behavior of the data link is writable.

135.1-2023 - 7.3.2.46.1.1 - Configure Network Through Network Port Object Test		
	Test Conditionality	Must be executed.
	Test Directives	Perform at least once. Repeat each time the network is reconfigured for a test. Execute test 7.3.2.46.1.1 at least once with ReinitializeDevice WARMSTART and at least once with ACTIVATE_CHANGES.
	Testing Hints	

9.8 Data Link Layer - IPv6

9.8.5 Supports Configuration Through Network Port Object

The IUT supports full, or partial, configuration of the data link through the Network Port object.

…		
135.1-2023 - 7.3.2.46.1.1 - Configure Network Through Network Port Object Test		
	Test Conditionality	Must be executed.
	Test Directives	Execute this test at least once on each Network Port object that has Network_Type = IPV6 and contains writable properties. Execute test 7.3.2.46.1.1 at least once with ReinitializeDevice WARMSTART and at least once with ACTIVATE_CHANGES.
	Testing Hints	

9.9 Data Link Layer - Secure Connect

9.9.7 Supports Configuration Through Network Port Object

The IUT supports full, or partial, configuration of the data link through the Network Port object. Specifically, at least 1 property in the Network Port object which changes the behavior of the data link is writable.

135.1-2023 - 7.3.2.46.1.1 - Configure Network Through Network Port Object Test		
	Test Conditionality	Must be executed.
	Test Directives	Perform at least once. Repeat each time the network is reconfigured for a test. Execute test 7.3.2.46.1.1 at least once with ReinitializeDevice WARMSTART and at least once with ACTIVATE_CHANGES.
	Testing Hints	

Specified Test Changes

None

BTL-23.3 fix3-7: Must Understand for Unknown Data Options [BTLWG-1548]

Overview:

A node that receives an Encapsulated-NPDU with a valid Secure Path Data Option plus a 'Must Understand' = 1 Header Marker with an Option Type between 2 and 30 is supposed to drop the message or respond with an error.

A node that receives an Encapsulated-NPDU with a valid Secure Path Data Option plus a 'Must Understand' = 0 Header Marker with an Option Type between 2 and 30 is supposed to process the message.

Changes:

Checklist Changes

None

Test Plan Changes

9.9 Data Link Layer - Secure Connect

[Modify section 9.9.1 Base Requirements in 9.9 Data Link Layer - Secure Connect]
[Add at the end of Section]

9.9.1 Base Requirements

Base requirements must be met by any IUT that supports BACnet/Secure Connect.

…

BTL - 12.5.1.1.X1 - Must Understand Header Marker for Unknown Data Options		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

Specified Test Changes

[Add a new test into BTL Specified Tests]

12.5.1.1.X1 Must Understand Header Marker for Unknown Data Options
Reason for Change: No test exists for this functionality.

Purpose: Test if a node correctly processes a message that includes unknown data Data Options with and without 'Must Understand' Header Marker.

Test Concept: With the IUT connected to the BACnet/SC network, OD sends a ReadProperty Request with a set of header options marked as 'Must Understand' = 1 and a Data Option between 2 and 30 in addition to the 'Secure Path' Data Option. Verify that the IUT does not process the message. Repeat with an additional ReadProperty Request with the same Data Options but 'Must Understand' = 0. Verify that this message was processed.

Configuration Requirements: The IUT is connected to the BACnet/SC network as a node.

Test Steps:

1. TRANSMIT Encapsulated-NPDU,
   'Message ID' = (M1: any valid value),
   'Originating Virtual Address' = (OD's VMAC),
   -- 'Destination Virtual Address' absent
   'Destination Options' (absent or any valid value),
   'Data Options' = ({X'C1'}, -- (more, M.U., no len, opt_type = 1 (Secure Path))

{X’5E’}), -- (not more, M.U., no len, opt_type = 30 (unknown header option type))
'BACnet NPDU' = ReadProperty-Request,
'Object Identifier' = (the IUT's Device object),
'Property Identifier' = Object_Name

2. CHECK(that the IUT does not respond with ReadProperty-ACK)
3. TRANSMIT Encapsulated-NPDU,
'Message ID' = (M2: any valid value),
'Originating Virtual Address' = (OD's VMAC),
-- 'Destination Virtual Address' absent
'Destination Options' (absent or any valid value),
'Data Options' = ({X'C1'}, -- (more, M.U., no len, opt_type = 1 (Secure Path))
{X’1E’}), -- (not more, not M.U., no len, opt_type = 30 (unknown header option type))
'BACnet NPDU' = ReadProperty-Request,
'Object Identifier' = (the IUT's Device object),
'Property Identifier' = Object_Name
4. RECEIVE Encapsulated-NPDU,
'Message ID' = M2,
-- 'Originating Virtual Address' absent
'Destination Virtual Address' = (OD's VMAC),
'Destination Options' (absent or any valid value),
'Data Options' = ({X'41' or a list of valid header options including Secure Path}),
'BACnet NPDU' = ReadProperty-ACK,
'Object Identifier' = (the IUT's Device object),
'Property Identifier' = Object_Name,
'Property Value' = (the IUT's device object name)

BTL-23.3 fix3-8: Must Understand for Unknown data Options - Router [BTLWG-1549]

Overview:

A router that receives a Encapsulated-NPDU with a valid Secure Path Data Option plus a 'Must Understand' = 1 or 'Must Understand' = 0 Header Marker with an Option Type between 2 and 30 that is routed into a secure network the message is to be routed unchanged.

A router that receives a Encapsulated-NPDU with a valid Secure Path Data Option plus a 'Must Understand' = 1 or 'Must Understand' = 0 Header Marker with an Option Type between 2 and 30 that is routed into a non-secure network the message is to be routed with the data attributes dropped.

Changes:

Checklist Changes

None

Test Plan Changes

10.1 Network Management - Routing

[Modify section 10.1.1 Base Requirements in 10.1 Network Management - Routing]
[Add at the end of Section]

10.1.1 Base Requirements

Base requirements must be met by any IUT that supports BACnet/Secure Connect.

…

BTL - 10.2.X7 - Must Understand for Unknown Data Options Forwarding Test		
	Test Conditionality	If the IUT is not able to route between multiple networks which support data_attributes, this test shall be skipped.
	Test Directives	
	Testing Hints	As of PR 22, BACnet/SC is the only standard datalink which supports data_attributes
BTL - 10.2.X8 - Must Understand for Unknown Data Options Dropping Test		
	Test Conditionality	If the IUT is not able to route between multiple networks which support data_attributes, this test shall be skipped.
	Test Directives	
	Testing Hints	As of PR 22, BACnet/SC is the only standard datalink which supports data_attributes

Specified Test Changes

[Add a new test into BTL Specified Tests]

10.2.X7 Must Understand for Unknown Data Options Forwarding Test
Reason for Change: No test exists for this functionality.

Purpose: To verify that routers which connects multiple network supporting data attributes, correctly routes data attributes that include unknown data Options and a valid 'Secure Path' Data Option.

Test Concept: With the IUT configured as a router between two networks which supports data attributes (such as BACnet/SC), send to the router a message which needs to be routed to the next network, which contains unknown data option data attributes in addition to a valid ‘Secure Path’ Data Option. Verify that the message is correctly routed and the data attributes are included in the routed message.

Configuration Requirements: The IUT shall be configured as a router between 2 networks which support data attributes.

Test Steps:

1. TRANSMIT PORT A,
   DA = LOCAL BROADCAST,
   SOURCE = D1A,
   'Data Options' = ({X'C1' }, -- (more, M.U., no len, opt_type = 1 (Secure Path))
   { X'5E' }), -- (not more, M.U., no len, opt_type = 30 (unknown header option type)),
   DNET = GLOBAL BROADCAST,
   DLEN = 0,
   Hop Count = 255,
   BACnet-Unconfirmed-Request-PDU,
   'Service Choice' = Who-Is
2. RECEIVE PORT B,
   DA = LOCAL BROADCAST,
   SA = IUT,
   'Data Options' = ({ X'C1' }, -- (more, M.U., no len, opt_type = 1 (Secure Path))
   { X'5E' }), -- (not more, M.U., no len, opt_type = 30 (unknown header option type)),
   DNET = GLOBAL BROADCAST,
   DLEN = 0,
   SNET = 1,
   SADR = D1A,
   Hop Count = (any integer x: 0 < x < 255),
   BACnet-Unconfirmed-Request-PDU,
   'Service Choice' = Who-Is
3. TRANSMIT PORT A,
   DA = LOCAL BROADCAST,
   SOURCE = D1A,
   'Data Options' = ({ X'C1' }, -- (more, M.U., no len, opt_type = 1 (Secure Path))
   { X'1E' }), -- (not more, no M.U., no len, opt_type = 30 (unknown header option type)),
   DNET = GLOBAL BROADCAST,
   DLEN = 0,
   Hop Count = 255,
   BACnet-Unconfirmed-Request-PDU,
   'Service Choice' = Who-Is
4. RECEIVE PORT B,
   DA = LOCAL BROADCAST,
   SA = IUT,
   'Data Options' = ({ X'C1' }, -- (more, M.U., no len, opt_type = 1 (Secure Path))
   { X'1E' }), -- (not more, no M.U., no len, opt_type = 30 (unknown header option type)),
   DNET = GLOBAL BROADCAST,
   DLEN = 0,
   SNET = 1,
   SADR = D1A,
   Hop Count = (any integer x: 0 < x < 255),
   BACnet-Unconfirmed-Request-PDU,
   'Service Choice' = Who-Is

10.2.X8 Must Understand for Unknown Data Options Dropping Test
Reason for Change: No test exists for this functionality.

Purpose: To verify that routers correctly drop data attributes that include unknown Data Options and a valid ‘Secure Path’ Data Option.

Test Concept: With the IUT configured as a router from a network which support data_attributes (such as BACnet/SC) to a network which does not support data_attributes (such as BACnet/IP), send to the router a message which needs to be routed to

the next network, and which contains data_attributes that include unknown data Options and Must Understand = TRUE. Verify that message is correctly routed and the data_attributes are silently dropped. Repeat with Must Understand = FALSE and verify the message is correctly routed and the data_attributes are silently dropped.

Configuration Requirements: The IUT shall be configured as a router between 2 networks in which the destination network does not support data attributes.

Test Steps:

1. TRANSMIT PORT A,
   DA = LOCAL BROADCAST,
   SOURCE = D1A,
   'Data Options' = ({ X'C1' }, -- (more, M.U., no len, opt_type = 1 (Secure Path))
   { X'5E' }), -- (not more, M.U., no len, opt_type = 30 (unknown header option type)),
   DNET = GLOBAL BROADCAST,
   DLEN = 0,
   Hop Count = 255,
   BACnet-Unconfirmed-Request-PDU,
   'Service Choice' = Who-Is
2. RECEIVE PORT B,
   DA = LOCAL BROADCAST,
   SA = IUT,
   DNET = GLOBAL BROADCAST,
   DLEN = 0,
   SNET = 1,
   SADR = D1A,
   Hop Count = (any integer x: 0 < x < 255),
   BACnet-Unconfirmed-Request-PDU,
   'Service Choice' = Who-Is
3. TRANSMIT PORT A,
   DA = LOCAL BROADCAST,
   SOURCE = D1A,
   'Data Options' = ({ X'C1' }, -- (more, M.U., no len, opt_type = 1 (Secure Path))
   { X'1E' }), -- (not more, no M.U., no len, opt_type = 30 (unknown header option type)),
   DNET = GLOBAL BROADCAST,
   DLEN = 0,
   Hop Count = 255,
   BACnet-Unconfirmed-Request-PDU,
   'Service Choice' = Who-Is
4. RECEIVE PORT B,
   DA = LOCAL BROADCAST,
   SA = IUT,
   DNET = GLOBAL BROADCAST,
   DLEN = 0,
   SNET = 1,
   SADR = D1A,
   Hop Count = (any integer x: 0 < x < 255),
   BACnet-Unconfirmed-Request-PDU,
   'Service Choice' = Who-Is

BTL-23.3 fix3-9: Must Understand for Unknown Data Options - Hub [BTLWG-1561]

Overview:

A B/SC hub that receives an Encapsulated-NPDU with a valid Secure Path Data Option plus a 'Must Understand' = 1 or 'Must Understand' = 0 Header Marker with an Option Type between 2 and 30 must forward the message unchanged.

Changes:

Checklist Changes

None

Test Plan Changes

10.1 Network Management - Routing

[Modify section 9.9.3]
[Add at the end of Section]

9.9.3 Is Able to Operate as a Hub

The IUT is able to operate as a hub (contains a hub function).

…

BTL - 12.5.2.1.X1 - Must Understand for Unknown Data Options Local Broadcast Execution Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 12.5.2.1.X2 - Must Understand for Unknown Data Options Forwards Unicast BVLCs Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

Specified Test Changes

[Add new tests into BTL Specified Tests]

12.5.2.1.X1 Must Understand for Unknown Data Options Local Broadcast Execution Test

Purpose: To verify that IUT, as a hub, correctly accepts and distributes broadcast messages that contain an unknown Header Type.

Test Concept: With the IUT operating as a hub, D4 sends a broadcast message with unknown header type and Must Understand = TRUE to the hub. Verify that the message is forwarded to all hub connectors except D4. D4 sends a broadcast message with unknown header type and Must Understand = FALSE to the hub. In both cases, verify the messages are forwarded to all hub connectors except D4.

Configuration Requirements: The IUT is operating as a hub and devices D2, D3, and D4 are connected to it.

Notes to Tester: The order of the broadcasts sent by the hub and the I-Am response can be sent in any order.

Test Steps:

-- Must Understand = 1

```
1.  TRANSMIT PORT (D4-IUT hub WebSocket),
        Encapsulated-NPDU,
        -- 'Originating Virtual Address' absent
        'Destination Virtual Address' =      X'FFFFFFFFFFFF', -- the local broadcast VMAC
        -- 'Destination Options' absent
        'Data Options' = ({X'C1'}, -- (more, M.U., no len, opt_type = 1 (Secure Path))
                         {X'5E'}), -- (not more, M.U., no len, opt_type = 30 (unknown header option type)),
        'Payload'
            Who-Is-Request
2.  REPEAT Dx = (D2, D3) DO {
        RECEIVE PORT (Dx-IUT hub WebSocket),
            Encapsulated-NPDU,
            'Originating Virtual Address' =(D4's VMAC),
            'Destination Virtual Address' =X'FFFFFFFFFFFF',          -- the local broadcast VMAC
            -- 'Destination Options' absent
        'Data Options' = ({X'C1'}, -- (more, M.U., no len, opt_type = 1 (Secure Path))
                         {X'5E'}), -- (not more, M.U., no len, opt_type = 30 (unknown header option type)),
            'Payload'
                Who-Is-Request
    }
3.  RECEIVE PORT (D4-IUT hub WebSocket),
        Encapsulated-NPDU,
        'Originating Virtual Address' =      (IUT's VMAC)
        'Destination Virtual Address' =      (D4's VMAC or X'FFFFFFFFFFFF', the local broadcast VMAC)
        -- 'Destination Options' absent
        'Data Options' = ({X'C1'}, -- (more, M.U., no len, opt_type = 1 (Secure Path))
                         {X'5E'}), -- (not more, M.U., no len, opt_type = 30 (unknown header option type)),
        'Payload'
            I-Am-Request,
            'I Am Device Identifier' =       (the IUT's Device object),
            'Max APDU Length Accepted' =     (the value specified in the EPICS),
            'Segmentation Supported' =       (the value specified in the EPICS),
            'Vendor Identifier' =   (the identifier registered for this vendor)
-- Must Understand = 0
4.  TRANSMIT PORT (D4-IUT hub WebSocket),
        Encapsulated-NPDU,
        -- 'Originating Virtual Address' absent
        'Destination Virtual Address' =      X'FFFFFFFFFFFF', -- the local broadcast VMAC
        -- 'Destination Options' absent
        'Data Options' = ({X'C1'}, -- (more, M.U., no len, opt_type = 1 (Secure Path))
                         {X'1E'}), -- (not more, no M.U., no len, opt_type = 30 (unknown header option type)),
        'Payload'
            Who-Is-Request
5.  REPEAT Dx = (D2, D3) DO {
        RECEIVE PORT (Dx-IUT hub WebSocket),
            Encapsulated-NPDU,
            'Originating Virtual Address' =(D4's VMAC),
            'Destination Virtual Address' =X'FFFFFFFFFFFF',          -- the local broadcast VMAC
            -- 'Destination Options' absent
        'Data Options' = ({X'C1'}, -- (more, M.U., no len, opt_type = 1 (Secure Path))
                         {X'1E'}), -- (not more, no M.U., no len, opt_type = 30 (unknown header option type)),
            'Payload'
                Who-Is-Request
    }
6.  RECEIVE PORT (D4-IUT hub WebSocket),
        Encapsulated-NPDU,
        'Originating Virtual Address' =      (IUT's VMAC)
        'Destination Virtual Address' =      (D4's VMAC or X'FFFFFFFFFFFF', the local broadcast VMAC)
        -- 'Destination Options' absent
        'Data Options' = ({X'C1'}, -- (more, M.U., no len, opt_type = 1 (Secure Path))
```

{X'1E'}), -- (not more, no M.U., no len, opt_type = 30 (unknown header option type)),
'Payload'
I-Am-Request,
'I Am Device Identifier' = (the IUT's Device object),
'Max APDU Length Accepted' = (the value specified in the EPICS),
'Segmentation Supported' = (the value specified in the EPICS),
'Vendor Identifier' = (the identifier registered for this vendor)

12.5.2.1.X2 Must Understand for Unknown Data Options Forwards Unicast BVLCs Test

Purpose: To verify that a hub correctly forwards unicast BVLCs received that contain an unknown Header Type.

Test Concept: The IUT is operating as the primary hub. D3 sends a unicast message with unknown header type and Must Understand = TRUE to D4 via the hub. Verify that the IUT correctly forwards the message to D4. D3 sends a unicast message with unknown header type and Must Understand = FALSE to D4 via the hub. Verify that the IUT correctly forwards the message to D4.

Configuration Requirements: The IUT is configured as the primary hub. D3 and D4 are connected to the IUT's hub function.

Test Steps:

-- verify that D3 and D4 can communicate with each other
-- Must Understand = 1
1. TRANSMIT PORT (D3-IUT hub WebSocket),
Encapsulated-NPDU,
'Message ID' = (M1: any valid value),
-- 'Originating Virtual Address' absent
'Destination Virtual Address' = (D4's VMAC),
-- 'Destination Options' absent
'Data Options' = ({X'C1'}, -- (more, M.U., no len, opt_type = 1 (Secure Path))
{X'5E'}), -- (not more, M.U., no len, opt_type = 30 (unknown header option type)),
'Payload' =
Who-Is-Request
2. RECEIVE PORT (D4-IUT hub WebSocket),
Encapsulated-NPDU,
'Message ID' = (M1: any valid value),
'Originating Virtual Address' = (D3's VMAC),
-- 'Destination Virtual Address' absent
-- 'Destination Options' absent
'Data Options' = ({X'C1'}, -- (more, M.U., no len, opt_type = 1 (Secure Path))
{X'5E'}), -- (not more, M.U., no len, opt_type = 30 (unknown header option type)),
'Payload' =
Who-Is-Request
-- Must Understand = 0
3. TRANSMIT PORT (D3-IUT hub WebSocket),
Encapsulated-NPDU,
'Message ID' = (M1: any valid value),
-- 'Originating Virtual Address' absent
'Destination Virtual Address' = (D4's VMAC),
-- 'Destination Options' absent
'Data Options' = ({X'C1'}, -- (more, M.U., no len, opt_type = 1 (Secure Path))
{X'1E'}), -- (not more, no M.U., no len, opt_type = 30 (unknown header option type)),
'Payload' =
Who-Is-Request
4. RECEIVE PORT (D4-IUT hub WebSocket),
Encapsulated-NPDU,
'Message ID' = (M1: any valid value),
'Originating Virtual Address' = (D3's VMAC),
-- 'Destination Virtual Address' absent
-- 'Destination Options' absent

```
'Data Options' = ({X'C1'}, -- (more, M.U., no len, opt_type = 1 (Secure Path))
                  {X'1E'}), -- (not more, no M.U., no len, opt_type = 30 (unknown header option type)),
'Payload' =
      Who-Is-Request
```