



BACnet[®] TESTING LABORATORIES ADDENDA

Addendum fix1 to BTL Test Package 26.0

**Revision final
Revised 5/27/2025**

Approved by the BTL Working Group on March 20, 2025;
Approved by the BTL Working Group Voting Members on May 23, 2025;
Published on May 29, 2025.

[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-26.0 fix1-1: Value Source Improvements [BTLWG-1590].....2

BTL-26.0 fix1-2: Fix SC_Hub_Function_Enable Property Test [BTLWG-1644].....8

BTL-26.0 fix1-3: Fix Heartbeat-Request Initiation Test [BTLWG-1648] 11

BTL-26.0 fix1-4: Fix Configures Other Device’s MAC Address Test [BTLWG-1660]..... 13

BTL-26.0 fix1-5: Network Port Object Testing Changes [BTLWG-1676] 15

In the following document, language to be added to existing clauses within the BTL Test Package 26.0 is indicated through the use of *italics*, while deletions are indicated by ~~strike through~~. 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-26.0 fix1-1: Value Source Improvements [BTLWG-1590]**Overview:**

The present Value Source tests infer that a BACnet Address can be written to the Value_Source property. Only the Device or Device/Object can be written. See Clause 19.5 last paragraph, Clause 19.5.1.3, paragraph 2. Some of the tests do not test all the value source properties. Testing of locally generated changes to the Present_Value is not tested. Clause 19.5.1.3, Paragraph 3.

Changes:**Checklist Changes**

None

Test Plan Changes

[Change 3.x.x Supports the Value Source Mechanism]

[Clauses 3.2.4, 3.6.9, 3.15.7, 3.39.4, 3.40.4, 3.42.10, 3.53.20, 3.54.9, 3.55.8]

3.x.x Supports the Value Source Mechanism

...

BTL - 7.3.1.28.3 - Value Source Property None Test		
	Test Conditionality	Must be executed
	Test Directives	
	Testing Hints	
BTL - 7.3.1.28.4 - Commandable Value Source Test		
	Test Conditionality	Must be executed
	Test Directives	
	Testing Hints	
BTL - 7.3.1.28.1 - Writing to the Value_Source Property by a Device Other than the Device that Commanded the Object		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 7.3.1.28.X1 - Value Source Initiated Locally Test		
	Test Conditionality	Must be executed if the Present_Value of an object in the device can be changed by an object or some local mechanism in the device.
	Test Directives	
	Testing Hints	

[Change 3.y.y Supports the Value Source Mechanism]

[Clauses 3.3.4, 3.7.8, 3.16.7, 3.24.4, 3.25.4, 3.26.6, 3.27.6, 3.28.6, 3.29.6, 3.30.4, 3.31.4, 3.32.4, 3.33.4, 3.34.6, 3.35.6]

3.y.y Supports the Value Source Mechanism

...

BTL - 7.3.1.28.2 - Non-commandable Value Source Property Test		
	Test Conditionality	Must be executed if the Value Source Mechanism is supported in a instance where Present_Value is not commandable
	Test Directives	
	Testing Hints	
BTL - 7.3.1.28.3 - Value_Source Property None Test		

	Test Conditionality	Must be executed if the Value Source Mechanism is supported in an instance where Present Value is commandable
	Test Directives	
	Testing Hints	
BTL - 7.3.1.28.4 - Commandable Value Source Test		
	Test Conditionality	Must be executed if the Value Source Mechanism is supported in an instance where Present Value is commandable
	Test Directives	
	Testing Hints	
BTL - 7.3.1.28.1 - Writing to the Value_Source Property by a Device Other than the Device that Commanded the Object		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 7.3.1.28.X1 - Value Source Initiated Locally Test		
	Test Conditionality	Must be executed if the Present Value of an object in the device can be changed by an object or some local mechanism in the device.
	Test Directives	
	Testing Hints	

[Change 3.z.z Supports the Value Source Mechanism]

[Clauses 3.9.6, 3.43.6, 3.53.20, 3.62.6]

3.z.z Supports the Value Source Mechanism

...

BTL - 7.3.1.28.2 - Non-commandable Value_Source Property Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 7.3.1.28.1 - Writing to the Value_Source Property by a Device Other than the Device that Commanded the Object		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 7.3.1.28.X1 - Value Source Initiated Locally Test		
	Test Conditionality	Must be executed if the Present Value of an object in the device can be changed by an object or some local mechanism in the device.
	Test Directives	
	Testing Hints	

[Change 3.a.a Supports the Value Source Mechanism]

[Clauses 3.39.4, 3.40.4]

3.a.a Supports the Value Source Mechanism

...

BTL - 7.3.1.28.5 - Life Safety Value_Source Property Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 7.3.1.28.1 - Writing to the Value_Source Property by a Device Other than the Device that Commanded the Object		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

BTL - 7.3.1.28.X1 - Value Source Initiated Locally Test		
	Test Conditionality	Must be executed if the Present Value of an object in the device can be changed by an object or some local mechanism in the device.
	Test Directives	
	Testing Hints	

Specified Test Changes

7.3.1.28.1 Writing to the Value_Source Property by a Device Other than the Device that Commanded the Object

Reason for Change: Test does not check the value source properties for a commandable object. PRIO is not defined.

Purpose: To verify the IUT correctly refuses an attempt to write a Value_Source property by a device other than the device that most recently commanded the object.

Test Concept: Command an object, O1, that supports the value source mechanism, from device D1 (TD), and verify the updated Value_Source *and, if commandable, the other value source properties*. Attempt to write to the Value_Source property from device D2. Verify that an error is returned and Value_Source *and the other value source properties do does* not change.

Configuration Requirements: If commandable, the value selected for PRIO shall be numerically small enough to be the active priority.

Notes to Tester: The value of the network-number in the network address of the Value_Source property will be one of three values. It will be 0 if the TD and IUT are on the same network and the IUT does not know the network number.

It will be the network number of the local network if the IUT knows the network number and the TD and IUT are on the same network. It will be the TD's network number if the TD is on a different network.

Test Steps:

1. TRANSMIT WriteProperty-Request,
SOURCE = D1,
'Object Identifier' = O1,
'Property Identifier' = (P1: the property monitored by the Value_Source mechanism for this object type),
'Priority' = (*absent or PRIO; absent or any value other than 6*)
'Property Value' = (X2: any valid value)
2. RECEIVE BACnet-SimpleACK-PDU
3. *VERIFY Value_Source = (VS, D1's device identifier or network address)*
- ~~4.~~ IF (O1 is commandable) THEN
 5. VERIFY Priority Array = X2, ARRAY_INDEX = PRIO
 6. *VERIFY Value_Source Array = VS, ARRAY_INDEX = PRIO*
 7. *VERIFY Last_Command_Time ~= (T1, the current local time)*
 8. *IF (Command_Time_Array is present) THEN*
 9. *VERIFY Command_Time_Array = T1, ARRAY_INDEX = PRIO*
- ELSE
 10. VERIFY (P1) = X2
 - ~~4.~~ *VERIFY Value_Source = (D1's device identifier or network address)*
-- Attempt to write D1's device identifier from D2
 - ~~11~~5. TRANSMIT WriteProperty-Request,
SOURCE = D2,
'Object Identifier' = O1,
'Property Identifier' = Value_Source,
'Priority' = PRIO,
'Property Value' = (any valid value)
 - ~~12~~6. RECEIVE BACnet-Error PDU,
'Error Class' = PROPERTY,
'Error Code' = WRITE_ACCESS_DENIED
-- Verify value source properties are unchanged

137. VERIFY Value_Source = (VSD1's device identifier or network address)

14. IF (O1 is commandable) THEN

15. VERIFY Value_Source Array = VS, ARRAY_INDEX = PRIO

16. VERIFY Last_Command_Time = T1

17. IF (Command_Time_Array is present) THEN

18. VERIFY Command_Time_Array = T1, ARRAY_INDEX = PRIO

7.3.1.28.2 Non-commandable Value_Source Property Test

Reason for Change: Specify what is supposed to be written to the Value_Source property.

Purpose: To verify that the Value_Source property indicates the source of the current Present_Value in a non-commandable object.

Test Concept: Select a non-commandable object with a writable Present_Value which supports the Value Source mechanism. The TD writes the Present_Value is written, and it is verified that Value_Source is updated appropriately. Value_Source is then written to verify that the last writer, TD, can update it.

Notes to Tester: The value of the network-number in the network address of the Value_Source property will be one of three values. It will be 0 if the TD and IUT are on the same network and the IUT does not know the network number. It will be the network number of the local network if the IUT knows the network number and the TD and IUT are on the same network. It will be the TD's network number if the TD is on a different network.

Test Steps:

1. WRITE Present_Value = (V1, any valid value)
2. VERIFY Present_Value = V1
3. VERIFY Value_Source = (TD's device identifier or network address)
4. WRITE Value_Source = (VS(TD's device identifier, O1, any valid object identifier, any valid value, V2))
5. VERIFY Value_Source = VSV2

7.3.1.28.3 Value_Source Property None Test

Reason for Change: The other Value_Source properties are not tested. The minimum on/off requirement from 19.5.1.3 paragraph 4 is not tested.

Purpose: To verify that the Value_Source property shall have the value 'None' when there is no active value source.

Test Concept: If there is no active value source, i.e., the Present_Value has taken on the value of Relinquish_Default, then the Value_Source property shall have the value 'None' and the other value source properties are updated. If minimum on/off is supported verify the Value_Source_Array at priority 6 is commanded object, O1, and the Command_Time_Array has the present local time.

Configuration Requirements: The object (O1) to be tested shall have 1 non-NULL entry in its Priority_Array and the Current_Command_Priority has a value other than NULL or 6.

Test Steps:

1. READ PRIO = Current_Command_Priority
2. ~~CHECK(PRIO < 6 and PRIO < NULL)~~
2. VERIFY Value_Source = (is not 'None')
3. WRITE Present_Value = NULL, PRIORITY = PRIO
4. VERIFY Last_Command_Time ~ (T1, the current local time)
5. VERIFY Value_Source_Array = TD, ARRAY_INDEX = PRIO
6. IF (Command_Time_Array is present) THEN
7. VERIFY Command_Time_Array = T1, ARRAY_INDEX = PRIO
8. IF (O1 has Minimum_On_Time or Minimum_Off_Time properties) THEN
9. WAIT the larger of Minimum_Off_Time and Minimum_On_Time
10. VERIFY Value_Source_Array = O1, ARRAY_INDEX = 6
11. IF (Command_Time_Array is present) THEN

12. *VERIFY Command_Time_Array ~=(the current local time), ARRAY_INDEX = 6*

13. VERIFY Current_Command_Priority = NULL

14. VERIFY Value_Source = 'None' -- the value is the choice 'none'

7.3.1.28.4 Commandable Value Source Test

Reason for Change: Command_Time_Array was not tested. Specify what is supposed to be written to the Value_Source property.

Purpose: To verify that the Value_Source, Value_Source_Array, and Last_Command_Time update correctly when Present_Value is written in a commandable object.

Test Concept: A commandable object which supports the value source mechanism is selected for the test. The Present_Value is written. Last_Command_Time, Value_Source, ~~and Value_Source_Array~~, *and Command_Time_Array* properties are checked to verify that they have been updated appropriately. Value_Source is then written, and it is verified that Last_Command_Time property has not changed.

Configuration Requirements: The object being tested shall be commandable and support the Value Source mechanism. *The value selected for PRIO shall be numerically small enough to be the active priority. No other internal processes shall be controlling the object.*

Notes to Tester: The value of the network-number in the network address of the Value_Source property will be one of three values. It will be 0 if the TD and IUT are on the same network and the IUT does not know the network number.

It will be the network number of the local network if the IUT knows the network number and the TD and IUT are on the same network. It will be the TD's network number if the TD is on a different network.

Test Steps:

-- Verify that the value source properties are updated when Present_Value is commanded.

1. WRITE Present_Value = (V1: any valid value), PRIORITY = (PRIO: ~~any value other than 6~~)

2. VERIFY Value_Source_Array = (~~VSSRC1~~: TD's device identifier or network address), ARRAY_INDEX = PRIO

3. VERIFY Value_Source = ~~VSSRC1~~

4. VERIFY Last_Command_Time ~=(~~T1~~, the current local time)

5. *IF (Command_Time_Array is present) THEN*

6. *VERIFY Command_Time_Array = T1, ARRAY_INDEX = PRIO*

-- Verify that Value_Source can be written and that Last_Command_Time does not update.

5. ~~READ T1 = (O1), Last_Command_Time~~

7. WRITE Value_Source = (~~VS(TD's device identifier, O1, any valid object identifier-SRC2: any valid value)~~), PRIORITY = PRIO

8. *VERIFY Value_Source = VS*

9. VERIFY Value_Source_Array = ~~VS-SRC2~~, ARRAY_INDEX = PRIO

8. ~~IF (Current_Command_Priority == PRIO) THEN~~

~~VERIFY Value_Source = SRC2~~

10. VERIFY Last_Command_Time = T1

7.3.1.28.5 Life Safety Value_Source Property Test

Reason for Change: Specify what is supposed to be written to the Value_Source property.

Purpose: To verify that the Value_Source property indicates the source of the current Mode property in a life safety object.

Test Concept: Select a life safety object which supports the Value Source mechanism. Mode is written, and it is verified that Value_Source is updated appropriately. Value_Source is then written to verify that the last writer can update it.

Notes to Tester: The value of the network-number in the network address of the Value_Source property will be one of three values. It will be 0 if the TD and IUT are on the same network and the IUT does not know the network number.

It will be the network number of the local network if the IUT knows the network number and the TD and IUT are on the same network. It will be the TD's network number if the TD is on a different network.

Test Steps:

1. WRITE Mode = V1
2. VERIFY Mode = V1
3. VERIFY Value_Source = (TD's device identifier or network address)
4. WRITE Value_Source = *(VS(TD's device identifier, O1, any valid object identifier any valid value, V2))*
5. VERIFY Value_Source = *VS V2*

7.3.1.28.X1 Value Source Initiated Locally Test

Reason for Change: No test for this functionality.

Purpose: To verify that a change to the Present_Value initiated by the local device results in updates to the Value Source properties. This change to the Present_Value could be from a Program object, Schedule object or some mechanism internal to the device.

Test Concept: The Present_Value is changed by a mechanism local to the device. The Value_Source is verified and, if the object is commandable, Last_Command_Time, Value_Source_Array, and Command_Time_Array properties are checked to verify that they have been updated appropriately.

Configuration Requirements: If commandable, the value selected for PRIO shall be numerically small enough to be the active priority.

Test Steps:

1. MAKE (O1, Present_Value change by some mechanism local to the device. If the object is commandable, at a PRIORITY = PRIO)
2. IF (the mechanism is from an object, O2) THEN
3. VERIFY Value_Source = VS, (IUT's device identifier, O2)
4. ELSE
5. VERIFY Value_Source = VS, (IUT's device identifier)
6. IF (O1 is commandable) THEN
7. VERIFY Value_Source_Array = VS, ARRAY_INDEX = PRIO
8. VERIFY Last_Command_Time ~= (T1, the current local time)
9. IF (Command_Time_Array is present) THEN
10. VERIFY Command_Time_Array = T1, ARRAY_INDEX = PRIO

BTL-26.0 fix1-2: Fix SC_Hub_Function_Enable Property Test [BTLWG-1644]

Overview:

According to Step 11 (11. CHECK that the IUT opens a WebSocket with the TD)in the test and the fact that the test shows the TD is the failover hub for the IUT at this point, the next two steps (12 & 13) should involve a connection request from the IUT to the TD, and acceptance from the TD as the hub. However, the current wording of these steps is the opposite: Step 12 currently says that the TD should send the connect request to the IUT, which is incorrect. The IUT should be the one initiating the connect request to the TD.

*. ASHRAE Standard 135 :

For hub connections to the BACnet/SC hub function, the hub connector of a BACnet/SC node is the initiating peer, and the BACnet/SC hub function is the accepting peer.

Changes:

Checklist Changes

None

Test Plan Changes

None

Specified Test Changes

12.5.2.1.X1 SC_Hub_Function_Enable Property Test

Reason for change: No Test for this functionality.

Reference: Addendum cc Clause 12.56.Y14.

Purpose: To ensure the IUTs hub function can be enabled and disabled using the SC_Hub_Function_Enable property.

Test Concept: With the IUTs SC_Hub_Function_Enable property set to TRUE, verify the IUT is operating as a hub. Change the IUTs SC_Hub_Function_Enable property to FALSE and verify the IUT is no longer operating as a hub.

Configuration Requirements: The IUT is configured as the primary hub and the value of the SC_Hub_Function_Enable property to TRUE. The TD is configured as the failover hub. The TDs primary hub URI is configured to reference the IUT, and the IUTs failover hub URI is configured to reference the TD.

Test Steps:

1. MAKE(the TD open a WebSocket to the IUT's hub function)
2. TRANSMIT PORT (TD-IUT hub WebSocket)

Connect-Request,

'Message ID' = (M1: any valid value),

-- 'Originating Virtual Address' absent

-- 'Destination Virtual Address' absent

-- 'Destination Options' absent

-- 'Data Options' absent

'VMAC Address' = (TD's VMAC),

'Device UUID' = (TD's UUID),

'Maximum BVLC Length' = (the TD's maximum BVLC accepted length),

'Maximum NPDU Length' = (the TD's maximum NPDU accepted length)
3. RECEIVE PORT (TD-IUT hub WebSocket)

```

Connect-Accept,
'Message ID' = M1,
-- 'Originating Virtual Address' absent
-- 'Destination Virtual Address' absent
'VMAC Address' = (IUT's VMAC),
'Device UUID' = (IUT's UUID),
'Maximum BVLC Length' = (the IUT's maximum BVLC accepted length),
'Maximum NPDU Length' = (the IUT's maximum NPDU accepted length)
4. IF (SC_Hub_Function_Enable is writable) THEN
5.   WRITE SC_Hub_Function_Enable = FALSE
6.   TRANSMIT ReinitializeDevice-Request
      'Reinitialized State of Device' = WARMSTART | ACTIVATE_CHANGES
      'Password' = (any valid password)
7.   RECEIVE BACnet-SimpleACK-PDU
      ELSE
8.     MAKE (SC_Hub_Function_Enable = FALSE)
9.     WAIT Activate_Changes Fail Time
10.    CHECK(that the TD attempts and fails to open a WebSocket to the IUT)
11.    CHECK(that the IUT opens a WebSocket with the TD)
12.    RECEIVE PORT (IUT-TD failover hub WebSocket)
      Connect-Request,
      'Message ID' = (M1: any valid value),
      -- 'Originating Virtual Address' absent
      -- 'Destination Virtual Address' absent
      -- 'Destination Options' absent
      -- 'Data Options' absent
      'VMAC Address' = (IUT's VMAC),
      'Device UUID' = (IUT's UUID),
      'Maximum BVLC Length' = (the IUT's maximum BVLC accepted length),
      'Maximum NPDU Length' = (the IUT's maximum NPDU accepted length)
13.    TRANSMIT PORT (IUT-TD failover hub WebSocket)
      Connect-Accept,
      'Message ID' = M2,
      -- 'Originating Virtual Address' absent
      -- 'Destination Virtual Address' absent
      'VMAC Address' = (TD's VMAC),
      'Device UUID' = (TD's UUID),
      'Maximum BVLC Length' = (the TD's maximum BVLC accepted length),
      'Maximum NPDU Length' = (the TD's maximum NPDU accepted length)
14.    VERIFY (SC_Hub_Function_Enable = FALSE)
15.    IF (SC_Hub_Function_Enable is writable) THEN
16.      WRITE SC_Hub_Function_Enable = TRUE
17.      TRANSMIT ReinitializeDevice-Request
        'Reinitialized State of Device' = WARMSTART | ACTIVATE_CHANGES
        'Password' = (any valid password)
18.      RECEIVE BACnet-SimpleACK-PDU
        ELSE
19.        MAKE (SC_Hub_Function_Enable = TRUE)
20.        WAIT Activate_Changes Fail Time
21.        MAKE(the TD open a WebSocket to the IUT's hub function)
22.        TRANSMIT PORT (TD-IUT hub WebSocket)
          Connect-Request,
          'Message ID' = (M1: any valid value),
          -- 'Originating Virtual Address' absent
          -- 'Destination Virtual Address' absent
          -- 'Destination Options' absent
          -- 'Data Options' absent
          'VMAC Address' = (TD's VMAC),
          'Device UUID' = (TD's UUID),
          'Maximum BVLC Length' = (the TD's maximum BVLC accepted length),

```

'Maximum NPDU Length' = (the TD's maximum NPDU accepted length)

23. RECEIVE PORT (TD-IUT hub WebSocket)

Connect-Accept,

'Message ID' = M1,

-- 'Originating Virtual Address' absent

-- 'Destination Virtual Address' absent

'VMAC Address' = (IUT's VMAC),

'Device UUID' = (IUT's UUID),

'Maximum BVLC Length' = (the IUT's maximum BVLC accepted length),

'Maximum NPDU Length' = (the IUT's maximum NPDU accepted length)

BTL-26.0 fix1-3: Fix Heartbeat-Request Initiation Test [BTLWG-1648]

Overview:

The change to Step # 2 was missed when the test was added to TP 26 Specified Tests.

2. BEFORE 1/2 of IUT's heartbeat interval + 10s

Changes:

Checklist Changes

None

Test Plan Changes

None

Specified Test Changes

12.5.1.1.16 Heartbeat-Request Initiation Test

Reason for Change: modified per Addendum 135-2020cc-1.

Purpose: To verify that the device initiates heartbeats as per its config.

Test Concept: With the IUT connected to the BACnet/SC network, ~~send a ReadProperty request to the IUT every heartbeat interval / 2 seconds. Verify that the IUT does not initiate a Heartbeat-Request. Stop sending messages to the IUT.~~ Wait the IUT's configured heart-beat interval plus 10 seconds and verify that the IUT sent a Heartbeat-Request, ensuring that no BVLCs are sent to the IUT during that period. *If the IUT claims Protocol_Revision 24 or greater heartbeat interval is the Network Port object, SC_Heartbeat_Timeout property.*

Configuration Requirements: Place the IUT in a mode where it will not initiate requests for a period longer than the heartbeat interval (except for the heartbeat request). If the IUT does not support DM-DCC-B and cannot be otherwise configured to behave in this manner, this test shall be skipped.

Test Steps:

1. REPEAT N = (1..Z) {
 TRANSMIT Encapsulated-NPDU,
 'Message ID' = (M1: any valid value),
 'Originating Virtual Address' = (OVA: any valid value, including absent),
 -- 'Destination Virtual Address' absent
 'Destination Options' (absent or any valid value),
 'Data Options' = ({ X'41' }), -- Secure Path
 'BACnet NPDU' =
 ReadProperty-Request,
 'Object Identifier' = (the IUT's Device object),
 'Property Identifier' = Object_Name
 RECEIVE Encapsulated-NPDU,
 'Message ID' = M1,
 -- 'Originating Virtual Address' absent
 'Destination Virtual Address' = OVA,
 'Destination Options' (absent or any valid value),

Path}),

'Data Options' = ({{ X'41' or a list of valid header options including Secure

'BACnet NPDU' =

ReadProperty-ACK,

'Object Identifier' = (the IUT's Device object),

'Property Identifier' = Object_Name,

'Property Value' = (the IUT's device object name)

~~——— WAIT ½ of IUT's heartbeat interval~~

~~}~~

~~2. CHECK(that the IUT did not send a HeartBeat during step 1)~~

~~—— Since we already waited ½ of an heartbeat interval, only ½ of that interval is now given for the IUT to~~

~~—— generate a Heartbeat-Request~~

23. BEFORE ½ of IUT's heartbeat interval + 10s

RECEIVE Heartbeat-Request,

'Message ID' = (M2: any valid value),

-- 'Originating Virtual Address' absent

-- 'Destination Virtual Address' absent

'Destination Options' = (absent or any valid value),

-- 'Data Options' absent

34. TRANSMIT Heartbeat-ACK,

'Message ID' = M2,

-- 'Originating Virtual Address' absent

-- 'Destination Virtual Address' absent

'Destination Options' = (absent or any valid value),

-- 'Data Options' absent

BTL-26.0 fix1-4: Fix Configures Other Device’s MAC Address Test [BTLWG-1660]

Overview:

In this test, the variant should be tested where only the MAC address is configured via the You-Are service, but not the Device-Identifier. However, in the test, the Device-Identifier is present. The parameter also includes the TD as "(TD's Device object)."

As a result, we would have actually only changed the MAC address, but still assigned both parameters. However, this is not the purpose of the test. The standard clearly states that in the structure of the You-Are service, the two parameters "Device-Identifier" and "Device MAC Address" are conditional and not mandatory.

If the test were conducted in this form, the case where only the MAC address is assigned and nothing else would never be tested. Additionally, the test would be redundant to "8.X36.2 Configures Other Device's Object Instance Number and MAC Address," where both parameters are assigned.

Changes:

Checklist Changes

None

Test Plan Changes

None

Specified Test Changes

8.X36.1 Configures Other Device's MAC Address

Reason for Change: No test exists for this functionality.

Purpose: To verify the IUT can configure another device's MAC address using the You-Are service without relying on the Who-Am-I service.

Test Concept: The IUT configures TD with an appropriate MAC address without TD first sending a Who-Am-I.

Configuration Requirements: TD's Device object is configured with a known Device object instance number and no configured MAC address.

Notes to Tester: The IUT may require the tester to specify all the parameters needed to configure TD with You-Are, using the IUT's software. The destination address used by TD shall be selected such that the IUT will receive the messages.

Test Steps:

1. MAKE (the IUT configure TD)
2. RECEIVE
 - DESTINATION = LOCAL BROADCAST | GLOBAL BROADCAST | REMOTE BROADCAST,
 - You-Are Request,
 - 'Vendor Identifier' = (TD's Vendor_Identifier),
 - 'Model Name' = (TD's Model_Name),
 - 'Serial Number' = (TD's Serial_Number),
 - 'Device Identifier' = (TD's Device object),
 - 'Device MAC Address' = (an appropriate MAC address)
3. IF (TD is not an MS/TP subordinate node)
 - TRANSMIT

DESTINATION = IUT | LOCAL BROADCAST | GLOBAL BROADCAST | REMOTE BROADCAST,
I-Am Request,
'Device Identifier' = (TD's Device object)
'Max APDU Length Accepted' = (any valid value),
'Segmentation Supported' = (any valid value),
'Vendor Identifier' = (TD's Vendor_Identifier)

BTL-26.0 fix1-5: Network Port Object Testing Changes [BTLWG-1676]

Overview:

An informal Interpretation Request was submitted to the SSPC that asked two main questions about the Network Port object. The first clarified that a Network Port object at Protocol_Level = PROTOCOL that is not referenced by another Network Port object can exist in a device. The second clarified that data links that have no required properties at Protocol_Level = PHYSICAL do not require a Network Port object at that level and, in that case, the Network Port object at Protocol_Level = PROTOCOL can contain a Reference_Port = 4194303.

Changes:

Checklist Changes

None

Test Plan Changes

9 Data Link Layer

[Change 9.1.10, 9.2.5, 9.3.11, 9.4.4, 9.5.4, 9.6.4, 9.7.4, 9.8.8, 9.9.15, 9.12.4]

9.1.10 Supports Hierarchical Network Port Objects

The IUT contains a Network Port object with Network Type = MSTP, Protocol_Level = BACNET_APPLICATION, and supports a set of Network Port objects which form a hierarchy of Network Port objects.

Verify EPICS		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision >= 24.
	Test Directives	Verify that each hierarchical Network Port object contains only required and optional properties based on its Network Type and Protocol Level.
	Testing Hints	
BTL - 7.3.2.46.X.1 - Valid Hierarchy Chain - MSTP Test		
	Test Conditionality	Must be executed.
	Test Directives	Repeat this test on all Network Port objects with Protocol_Level = BACNET_APPLICATION, Network_Type = MSTP, and Reference_Port <> 4194303.
	Testing Hints	
BTL - 7.3.2.46.4.2 - Properties in Referenced Network Port Reflected in Top Network Port Object		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision < 24.
	Test Directives	
	Testing Hints	
135.1-2023 - 7.3.2.46.4.3 - Changes Reflected in Top Network Port Object		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision < 24 and supports writable properties in its Network Port hierarchies.
	Test Directives	
	Testing Hints	
135.1-2023 - 7.3.2.46.4.4 - Changes Reflected in Lower Network Port Objects		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision < 24 and supports writable properties in its Network Port hierarchies.
	Test Directives	
	Testing Hints	

9.2.5 Supports Hierarchical Network Port Objects

The IUT contains a Network Port object with Network Type = MSTP, Protocol_Level = BACNET_APPLICATION, and supports a set of Network Port objects which form a hierarchy of Network Port objects.

Verify EPICS		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision ≥ 24 .
	Test Directives	Verify that each hierarchical Network Port object contains only required and optional properties based on its Network_Type and Protocol_Level.
	Testing Hints	
BTL - 7.3.2.46.X.1 - Valid Hierarchy Chain - MSTP Test		
	Test Conditionality	Must be executed.
	Test Directives	Repeat this test on all Network Port objects with Protocol_Level = BACNET_APPLICATION, Network_Type = MSTP, and Reference_Port ≤ 4194303 .
	Testing Hints	
BTL - 7.3.2.46.4.2 - Properties in Referenced Network Port Reflected in Top Network Port Object		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision < 24 .
	Test Directives	
	Testing Hints	
135.1-2023 - 7.3.2.46.4.3 - Changes Reflected in Top Network Port Object		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision < 24 and supports writable properties in its Network Port hierarchies.
	Test Directives	
	Testing Hints	
135.1-2023 - 7.3.2.46.4.4 - Changes Reflected in Lower Network Port Objects		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision < 24 and supports writable properties in its Network Port hierarchies.
	Test Directives	
	Testing Hints	

9.3.11 Supports Hierarchical Network Port Objects

The IUT contains a Network Port object with Network Type = IPV4, Protocol_Level = BACNET_APPLICATION and supports a set of Network Port objects which form a hierarchy of Network Port objects.

Verify EPICS		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision ≥ 24 .
	Test Directives	Verify that each hierarchical Network Port object contains only required and optional properties based on its Network_Type and Protocol_Level.
	Testing Hints	
BTL - 7.3.2.46.X.2 - Valid Hierarchy Chain - IPvX Test		
	Test Conditionality	Must be executed.
	Test Directives	Repeat this test on all Network Port objects with Protocol_Level = BACNET_APPLICATION, Network_Type = IPV4, and Reference_Port ≤ 4194303 .
	Testing Hints	
BTL - 7.3.2.46.4.2 - Properties in Referenced Network Port Reflected in Top Network Port Object		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision < 24 .
	Test Directives	
	Testing Hints	
135.1-2023 - 7.3.2.46.4.3 - Changes Reflected in Top Network Port Object		

	Test Conditionality	Must be executed if the IUT claims Protocol Revision < 24 and supports writable properties in its Network Port hierarchies.
	Test Directives	
	Testing Hints	
135.1-2023 - 7.3.2.46.4.4 - Changes Reflected in Lower Network Port Objects		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision < 24 and supports writable properties in its Network Port hierarchies.
	Test Directives	
	Testing Hints	

9.4.4 Supports Hierarchical Network Port Objects

The IUT contains a Network Port object with Network Type = ZIGBEE, Protocol_Level = BACNET_APPLICATION and supports a set of Network Port objects which form a hierarchy of Network Port objects.

Verify EPICS		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision >= 24.
	Test Directives	Verify that each hierarchical Network Port object contains only required and optional properties based on its Network Type and Protocol Level.
	Testing Hints	
BTL - 7.3.2.46.X.3 - Valid Hierarchy Chain - Application and Physical Test		
	Test Conditionality	Must be executed.
	Test Directives	Repeat this test on all Network Port objects with Protocol_Level = BACNET_APPLICATION, Network_Type = ZIGBEE, and Reference_Port <> 4194303.
	Testing Hints	
BTL - 7.3.2.46.4.2 - Properties in Referenced Network Port Reflected in Top Network Port Object		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision < 24.
	Test Directives	
	Testing Hints	
135.1-2023 - 7.3.2.46.4.3 - Changes Reflected in Top Network Port Object		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision < 24 and supports writable properties in its Network Port hierarchies.
	Test Directives	
	Testing Hints	
135.1-2023 - 7.3.2.46.4.4 - Changes Reflected in Lower Network Port Objects		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision < 24 and supports writable properties in its Network Port hierarchies.
	Test Directives	
	Testing Hints	

9.5.4 Supports Hierarchical Network Port Objects

The IUT contains a Network Port object with Network Type = ETHERNET, Protocol_Level = BACNET_APPLICATION and supports a set of Network Port objects which form a hierarchy of Network Port objects.

Verify EPICS		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision >= 24.
	Test Directives	Verify that each hierarchical Network Port object contains only required and optional properties based on its Network Type and Protocol Level.
	Testing Hints	
BTL - 7.3.2.46.4X.3 - Valid Hierarchy Chain - Application and Physical Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

	Test Directives	Repeat this test on all Network Port objects with Protocol_Level = BACNET_APPLICATION, Network_Type = ETHERNET, and Reference_Port <> 4194303.
	Testing Hints	
BTL - 7.3.2.46.4.2 - Properties in Referenced Network Port Reflected in Top Network Port Object		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision < 24.
	Test Directives	
	Testing Hints	
135.1-2023 - 7.3.2.46.4.3 - Changes Reflected in Top Network Port Object		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision < 24 and supports writable properties in its Network Port hierarchies.
	Test Directives	
	Testing Hints	
135.1-2023 - 7.3.2.46.4.4 - Changes Reflected in Lower Network Port Objects		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision < 24 and supports writable properties in its Network Port hierarchies.
	Test Directives	
	Testing Hints	

9.6.4 Supports Hierarchical Network Port Objects

The IUT contains a Network Port object with Network Type = ARCNET, Protocol_Level = BACNET_APPLICATION and supports a set of Network Port objects which form a hierarchy of Network Port objects.

Verify EPICS		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision >= 24.
	Test Directives	Verify that each hierarchical Network Port object contains only required and optional properties based on its Network_Type and Protocol_Level.
	Testing Hints	
BTL - 7.3.2.46.X.3 - Valid Hierarchy Chain - Application and Physical Test		
	Test Conditionality	Must be executed.
	Test Directives	Repeat this test on all Network Port objects with Protocol_Level = BACNET_APPLICATION, Network_Type = ARCNET, and Reference_Port <> 4194303.
	Testing Hints	
BTL - 7.3.2.46.4.2 - Properties in Referenced Network Port Reflected in Top Network Port Object		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision < 24.
	Test Directives	
	Testing Hints	
135.1-2023 - 7.3.2.46.4.3 - Changes Reflected in Top Network Port Object		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision < 24 and supports writable properties in its Network Port hierarchies.
	Test Directives	
	Testing Hints	
135.1-2023 - 7.3.2.46.4.4 - Changes Reflected in Lower Network Port Objects		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision < 24 and supports writable properties in its Network Port hierarchies.
	Test Directives	
	Testing Hints	

9.7.4 Supports Hierarchical Network Port Objects

The IUT contains a Network Port object with Network Type = LONTALK, Protocol_Level = BACNET_APPLICATION and supports a set of Network Port objects which form a hierarchy of Network Port objects.

Verify EPICS		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision \geq 24.
	Test Directives	Verify that each hierarchical Network Port object contains only required and optional properties based on its Network_Type and Protocol_Level.
	Testing Hints	
BTL - 7.3.2.46.X.3 - Valid Hierarchy Chain - Application and Physical Test		
	Test Conditionality	Must be executed.
	Test Directives	Repeat this test on all Network Port objects with Protocol_Level = BACNET_APPLICATION, Network_Type = LONTALK, and Reference_Port \leq 4194303.
	Testing Hints	
BTL - 7.3.2.46.4.2 - Properties in Referenced Network Port Reflected in Top Network Port Object		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision $<$ 24.
	Test Directives	
	Testing Hints	
135.1-2023 - 7.3.2.46.4.3 - Changes Reflected in Top Network Port Object		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision $<$ 24 and supports writable properties in its Network Port hierarchies.
	Test Directives	
	Testing Hints	
135.1-2023 - 7.3.2.46.4.4 - Changes Reflected in Lower Network Port Objects		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision $<$ 24 and supports writable properties in its Network Port hierarchies.
	Test Directives	
	Testing Hints	

9.8.8 Supports Hierarchical Network Port Objects

The IUT contains a Network Port object with Network Type = IPV6, Protocol_Level = BACNET_APPLICATION and supports a set of Network Port objects which form a hierarchy of Network Port objects.

Verify EPICS		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision \geq 24.
	Test Directives	Verify that each hierarchical Network Port object contains only required and optional properties based on its Network_Type and Protocol_Level.
	Testing Hints	
BTL - 7.3.2.46.X.2 - Valid Hierarchy Chain - IPvX Test		
	Test Conditionality	Must be executed.
	Test Directives	Repeat this test on all Network Port objects with Protocol_Level = BACNET_APPLICATION, Network_Type = IPV6, and Reference_Port \leq 4194303.
	Testing Hints	
BTL - 7.3.2.46.4.2 - Properties in Referenced Network Port Reflected in Top Network Port Object		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision $<$ 24.
	Test Directives	
	Testing Hints	
135.1-2023 - 7.3.2.46.4.3 - Changes Reflected in Top Network Port Object		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision $<$ 24 and supports writable properties in its Network Port hierarchies.
	Test Directives	

	Testing Hints	
135.1-2023 - 7.3.2.46.4.4 - Changes Reflected in Lower Network Port Objects		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision < 24 and supports writable properties in its Network Port hierarchies.
	Test Directives	
	Testing Hints	

9.9.15 Supports Hierarchical Network Port Objects

The IUT contains a Network Port object with Network Type = SECURE_CONNECT, Protocol_Level = BACNET_APPLICATION and supports a set of Network Port objects which form a hierarchy of Network Port objects.

Verify EPICS		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision >= 24.
	Test Directives	Verify that each hierarchical Network Port object contains only required and optional properties based on its Network_Type and Protocol_Level.
	Testing Hints	
BTL - 7.3.2.46.X.4 - Valid Hierarchy Chain - Secure Connect Test		
	Test Conditionality	Must be executed.
	Test Directives	Repeat this test on all Network Ports object with Protocol_Level = BACNET_APPLICATION, Network_Type = SECURE_CONNECT, and Reference_Port <> 4194303.
	Testing Hints	

[change Clause 9.12.4]

9.12.4 Supports Hierarchical Network Port Objects

The IUT contains a Network Port object with Network Type = proprietary, Protocol_Level = NON_BACNET_APPLICATION and supports a set of Network Port objects which form a hierarchy of Network Port objects.

Verify EPICS		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision >= 24.
	Test Directives	Verify that each hierarchical Network Port object contains only required and optional properties based on its Network_Type and Protocol_Level.
	Testing Hints	
BTL - 7.3.2.46.X.5 - Valid Hierarchy Chain - Proprietary Test		
	Test Conditionality	Must be executed.
	Test Directives	Repeat this test on all Network Port objects with Protocol_Level = NON_BACNET_APPLICATION, Network_Type = <proprietary>, and Reference_Port <> 4194303.
	Testing Hints	
BTL - 7.3.2.46.4.2 - Properties in Referenced Network Port Reflected in Top Network Port Object		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision < 24.
	Test Directives	
	Testing Hints	
135.1-2023 - 7.3.2.46.4.3 - Changes Reflected in Top Network Port Object		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision < 24 and supports writable properties in its Network Port hierarchies.
	Test Directives	
	Testing Hints	
135.1-2023 - 7.3.2.46.4.4 - Changes Reflected in Lower Network Port Objects		
	Test Conditionality	Must be executed if the IUT claims Protocol Revision < 24 and supports writable properties in its Network Port hierarchies.

	Test Directives	
	Testing Hints	

[Change 9.1.11, 9.2.6, 9.3.12, 9.4.5, 9.5.5, 9.6.5, 9.7.5, 9.8.9, 9.9.16]

9.x.y Supports Non-hierarchical Network Port Objects

....

Verify EPICS		
	Test Conditionality	Must be executed.
	Test Directives	Verify IUT contains non-hierarchical Network Port objects with Protocol_Level equal to BACNET_APPLICATION for this Network_Type.
	Testing Hints	
Verify EPICS		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision < 24.
	Test Directives	Verify the Reference_Port is absent or equal to 4194303 for all non-hierarchical Network Port objects of this Network_Type.
	Testing Hints	
Verify EPICS		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision >= 24.
	Test Directives	Verify the Reference_Port is absent for all non-hierarchical Network Port objects of this Network_Type.
	Testing Hints	
Verify EPICS		
	Test Conditionality	Must be executed.
	Test Directives	Verify the Additional_Reference_Ports property is absent for all non-hierarchical Network Port objects of this Network_Type.
	Testing Hints	
Verify EPICS		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision >= 24.
	Test Directives	Verify each non-hierarchical Network Port object contains all required properties for this Network_Type.
	Testing Hints	
Verify EPICS		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision >= 24.
	Test Directives	Verify each non-hierarchical Network Port object contains only valid optional properties for this Network_Type.
	Testing Hints	

[Change 9.12.5]

9.12.5 Supports Non-hierarchical Network Port Objects

The IUT contains a Network Port object with Network Type = proprietary, Protocol_Level = NON_BACNET_APPLICATION, and supports non-hierarchical Network Port objects.

Verify EPICS		
	Test Conditionality	Must be executed.
	Test Directives	Verify IUT contains non-hierarchical Network Port objects with Protocol_Level equal to NON_BACNET_APPLICATION for this Network_Type.
	Testing Hints	
Verify EPICS		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision < 24.

	Test Directives	Verify the Reference_Port is absent or equal to 4194303 for all non-hierarchical Network Port objects of this Network_Type.
	Testing Hints	
Verify EPICS		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision >= 24.
	Test Directives	Verify the Reference_Port is absent for all non-hierarchical Network Port objects of this Network_Type.
	Testing Hints	
Verify EPICS		
	Test Conditionality	Must be executed.
	Test Directives	Verify the Additional_Reference_Ports property is absent for all non-hierarchical Network Port objects of this Network_Type.
	Testing Hints	
Verify EPICS		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision >= 24.
	Test Directives	Verify each non-hierarchical Network Port object contains all required properties for this Network_Type.
	Testing Hints	
Verify EPICS		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision >= 24.
	Test Directives	Verify each non-hierarchical Network Port object contains only valid optional properties for this Network_Type.
	Testing Hints	

Specified Test Changes

[Remove 7.3.2.46.4.1]

7.3.2.46.4.1 Valid Hierarchy Test

Reason for Change: The test no longer needs to test all NPOs at BACNET_APPLICATION as the test now referenced in each DLL. Modified per Addendum 135-2020ec 1.

Purpose: To verify that the set of network port objects in the IUT are organized in a valid hierarchy.

Test Concept: *Starting with the* Visit each Network Port object (NP) which represents a configured application layer port. Ensure that the top Network Port object has a Protocol_Level of (BACNET_APPLICATION or NON_BACNET_APPLICATION). Visit *visit* each Network Port object in the hierarchy ensuring that the Protocol_Level properties are valid.

Test Steps:

```

1. REPEAT NP = (object id of each Network Port object which has a Protocol_Level of
   BACNET_APPLICATION or NON_BACNET_APPLICATION) DO {
2.   REPEAT NPx = (object id of each Network Port object in NP's hierarchy) DO {
3.     PL = READ (Network Port, NPx), Protocol_Level
4.     IF PL is BACNET_APPLICATION or NON_BACNET_APPLICATION THEN
5.       ERROR Invalid Protocol_Level in child Network Port object
6.     IF PL is PHYSICAL THEN
7.       VERIFY (Network Port, NPx), Reference_Port = 4194303
8.     }
9.   }
10. }
11. REPEAT NPx = (object id of each Network Port object, Reference_Port in NP's hierarchy) DO {
12.   PL = READ (Network Port, NPx), Protocol_Level
13.   IF (PL is BACNET_APPLICATION or NON_BACNET_APPLICATION) THEN
14.     ERROR Invalid Protocol_Level in child Network Port object
15.   IF (PL is PHYSICAL) THEN

```

```

6. VERIFY (Network_Port, NPx), Reference_Port = 4194303
—}
7. IF (Protocol_Revision >= 24 and Additional_Reference_Ports is present) THEN {
8. IF (NP, Reference_Port property is not present) THEN
9. ERROR missing Reference_Port property
10. REPEAT (for each entry Network_Port object, Additional_Reference_Ports) DO {
11. REPEAT NPx = (object id of each Network_Port object, Additional_Reference_Ports in NP's hierarchy) DO {
12. PL = READ (Network_Port, NPx), Protocol_Level
13. IF PL is BACNET_APPLICATION or NON_BACNET_APPLICATION THEN
14. ERROR Invalid Protocol_Level in child Network_Port object
15. IF PL is PHYSICAL THEN
16. VERIFY (Network_Port, NPx), Additional_Reference_Ports = (empty list)
—}
—}

```

[Add 7.3.2.46.X and sub tests]

7.3.2.46.X Valid Hierarchy Tests

7.3.2.46.X.1 Valid Hierarchy Chain - MSTP Test

Reason for Change: Explicit test for this data link.

Purpose: To verify that a hierarchical chain of MSTP network port objects in the IUT is organized in a valid hierarchy of three protocol levels.

Test Concept: Starting with a hierarchical Network Port object (NP_APP) which represents a configured application layer port-(BACNET_APPLICATION), verify this port contains valid Network_Type, Protocol_Level, and Reference_Port. Visit each Network Port object in the hierarchy ensuring that the Network_Type, Protocol_Level, and Reference_Port are valid.

Notes to Tester: NPOs of Network_Type = MSTP are excluded from this test if they are at Protocol_Level = PROTOCOL and they are not referenced by another NPO. NPOs are also excluded from this test if they are at Protocol_Level = PHYSICAL and Network_Type = SERIAL and they are not referenced by another NPO.

Test Steps:

```

-- Verify Protocol_Level = BACNET_APPLICATION
1.  VERIFY NP_APP, Protocol_Level = BACNET_APPLICATION
2.  VERIFY NP_APP, Network_Type = MSTP
3.  VERIFY NP_APP, Reference_Port <> 4194303

-- Verify Protocol_Level = PROTOCOL
4.  RP = READ (NP_APP, Reference_Port)
5.  NP_PROT = (Network_Port, RP)
6.  VERIFY NP_PROT, Protocol_Level = PROTOCOL
7.  VERIFY NP_PROT, Network_Type = MSTP
8.  VERIFY NP_PROT, Reference_Port <> 4194303

-- Verify Protocol_Level = PHYSICAL
9.  RP = READ (NP_PROT, Reference_Port)
10. NP_PHY = (Network_Port, RP)
11. VERIFY NP_PHY, Protocol_Level = PHYSICAL
12. VERIFY NP_PHY, Network_Type = SERIAL
13. VERIFY NP_PHY, Reference_Port = 4194303

```

7.3.2.46.X.2 Valid Hierarchy Chain - IPvX Test

Reason for Change: Explicit test for this data link.

Purpose: To verify that a hierarchical chain of IPV4 or IPV6 network port objects in the IUT is organized in a valid hierarchy of protocol levels.

Test Concept: Starting with a hierarchical Network Port object (NP_APP) and Network_Type (NT) of IPV4 or IPV6 which represents a configured application layer port (BACNET_APPLICATION), verify this port contains valid Network_Type, Protocol_Level, and Reference_Port. Visit each Network Port object in the hierarchy ensuring that the Network_Type, Protocol_Level, and Reference_Port are valid.

Notes to Tester: NPOs of this Network_Type are excluded from this test if they are at Protocol_Level = PROTOCOL and they are not referenced by another NPO. NPOs are also excluded from this test if they are at Protocol_Level = PHYSICAL and Network_Type = ETHERNET and they are not referenced by another NPO.

Test Steps:

```
-- Verify Protocol_Level = BACNET_APPLICATION
1.  VERIFY NP_APP, Protocol_Level = BACNET_APPLICATION
2.  VERIFY NP_APP, Network_Type = NT
3.  RP = READ NP_APP, Reference_Port
4.  VERIFY RP <> 4194303

-- Verify required NPO at Protocol_Level = PROTOCOL
5.  NP_PROT = (Network Port, RP)
6.  VERIFY NP_PROT, Protocol_Level = PROTOCOL
7.  VERIFY NP_PROT, Network_Type = NT
8.  RP = READ NP_PROT, Reference_Port
9.  IF (RP <> 4194303) THEN
    {
      -- Check if Protocol_Level = PHYSICAL
10.   NPx = (Network Port, RP)
11.   IF (NPx, Protocol_Level = PHYSICAL) THEN
      -- PHYSICAL level. Could be ETHERNET or proprietary
12.     VERIFY NPx, Reference_Port = 4194303
    ELSE
      {
        -- Some number of NPOs at PROTOCOL level
        -- move down the chain of NPOs to the bottom
13.     WHILE (RP <> 4194303)
        {
14.       NPx = (Network Port, RP)
15.       VERIFY NPx, Protocol_Level <> (BACNET_APPLICATION OR NON_BACNET_APPLICATION)
          -- check that if PHYSICAL level it is the last NPO
16.       IF (NPx, Protocol_Level == PHYSICAL) THEN
17.         VERIFY NPx, Reference_Port = 4194303
18.         RP = READ NPx, Reference_Port
        }
      -- either the NPO is at PROTOCOL and does not reference another NPO or it is PHYSICAL
      -- If PHYSICAL, it could be ETHERNET or proprietary or SERIAL -- done
    }
  }
ELSE
  {
    -- NPO is at PROTOCOL and does not reference another NPO - done
  }
}
```

7.3.2.46.X.3 Valid Hierarchy Chain - Application and Physical Test

Reason for Change: Explicit test for data links without PROTOCOL level NPOs.

Purpose: To verify that a hierarchical chain of network port objects in the IUT is organized in a valid hierarchy of BACNET_APPLICATION and PHYSICAL protocol levels.

Test Concept: Starting with a hierarchical Network Port object (NP_APP) which represents a configured application layer port-(BACNET_APPLICATION), verify this port contains the correct Network_Type (NT), Protocol_Level, and Reference_Port. Visit the next level ensuring that the Network_Type, Protocol_Level, and Reference_Port are correct.

Notes to Tester: NPOs of this Network_Type with Protocol_Level = PHYSICAL are excluded from this test if they are not referenced by another NPO.

Test Steps:

```
-- Verify Protocol_Level = BACNET_APPLICATION
1.  VERIFY NP_APP, Protocol_Level = BACNET_APPLICATION
2.  VERIFY NP_APP, Network_Type = NT
3.  VERIFY NP_APP, Reference_Port <> 4194303

-- Verify Protocol_Level = PHYSICAL
4.  RP = READ (NP_APP, Reference_Port)
5.  NP_PHY = (Network Port, RP)
6.  VERIFY NP_PHY, Protocol_Level = PHYSICAL
7.  VERIFY NP_PHY, Network_Type = NT
8.  VERIFY NP_PHY, Reference_Port = 4194303
```

7.3.2.46.X.4 Valid Hierarchy Chain - Secure Connect Test

Reason for Change: Explicit test for this data link.

Purpose: To verify that a hierarchical chain of SECURE_CONNECT network port objects in the IUT is organized in a valid hierarchy of protocol levels.

Test Concept: Starting with a hierarchical Network Port object (NP_APP) and Network_Type of SECURE_CONNECT which represents a configured application layer port (BACNET_APPLICATION), verify this port contains valid Network_Type, Protocol_Level, and Reference_Port. Visit each Network Port object in the hierarchy ensuring that the Network_Type, Protocol_Level and Reference_Port are valid. Using the Additional_Reference_Ports property, visit each Network Port object in the hierarchy ensuring that the Network_Type, Protocol_Level, Reference_Port, and Additional_Reference_Ports are valid.

Test Steps:

```
-- Verify Protocol_Level = BACNET_APPLICATION
1.  VERIFY NP_APP, Protocol_Level = BACNET_APPLICATION
2.  VERIFY NP_APP, Network_Type = SECURE_CONNECT
3.  RP = READ NP_APP, Reference_Port
4.  VERIFY RP <> 4194303

-- Check the rest of the chain
5.  WHILE (RP <> 4194303)
    {
6.      NPx = (Network Port, RP)
7.      VERIFY NPx, Protocol_Level <> (NON_BACNET_APPLICATION or BACNET_APPLICATION)
8.      VERIFY NPx, Network_Type <> SECURE_CONNECT
      -- check that if PHYSICAL level it is the last NPO
9.      IF (NPx, Protocol_Level == PHYSICAL) THEN
10.         VERIFY NPx, Reference_Port = 4194303
11.         RP = READ NPx, Reference_Port
    }
    -- either the NPO is at PROTOCOL and does not reference another NPO or it is PHYSICAL

12. IF (NP_APP, Additional_Reference_Ports is present) THEN {
13.     REPEAT ARP = (for each entry in NP_APP, Additional_Reference_Ports) DO {
14.         NPx = (Network Port, ARP)
```

```

15.    VERIFY NPx, Protocol_Level <> (NON_BACNET_APPLICATION or BACNET_APPLICATION)
16.    VERIFY NPx, Network_Type <> SECURE_CONNECT
    -- check that if PHYSICAL level it is the last NPO
17.    IF (NPx, Protocol_Level == PHYSICAL) THEN
18.        IF (NPx, Additional_Reference_Ports is present) THEN
19.            VERIFY NPx, Additional_Reference_Ports = (empty list)
    -- Check the rest of the chain
20.    RP = READ NPx, Reference_Port
21.    WHILE (RP <> 4194303)
    {
22.        NPx = (Network Port, RP)
23.        VERIFY NPx, Protocol_Level <> (NON_BACNET_APPLICATION
    or BACNET_APPLICATION)
24.        VERIFY NPx, Network_Type <> SECURE_CONNECT
    -- check that if PHYSICAL level it is the last NPO
25.        IF (NPx, Protocol_Level == PHYSICAL) THEN
26.            VERIFY NPx, Reference_Port = 4194303
27.            RP = READ NPx, Reference_Port
    }
    }
}

```

7.3.2.46.X.5 Valid Hierarchy Chain - Proprietary Test

Reason for Change: Explicit test for this data link.

Purpose: To verify that a hierarchical chain of non-BACnet network port objects in the IUT is organized in a valid hierarchy of protocol levels.

Test Concept: Starting with a hierarchical Network Port object (NP_APP) and Network_Type (NT) which represents a configured application layer port (NON_BACNET_APPLICATION), verify this port contains valid Network_Type, Protocol_Level, and Reference_Port. Visit each Network Port object in the hierarchy ensuring that the Network_Type, Protocol_Level, and Reference_Port are valid. Using the Additional_Reference_Ports property, visit each Network Port object in the hierarchy ensuring that the Network_Type, Protocol_Level, Reference_Port, and Additional_Reference_Ports are valid.

Notes to Tester: NPOs of this Network_Type are excluded from this test if they are at Protocol_Level = PROTOCOL and they are not referenced by another NPO. NPOs of this Network_Type are also excluded from this test if they are at Protocol_Level = PHYSICAL and they are not referenced by another NPO.

Test Steps:

```

-- Verify Protocol_Level = NON_BACNET_APPLICATION
1.  VERIFY NP_APP, Protocol_Level = NON_BACNET_APPLICATION
2.  VERIFY NP_APP, Network_Type = NT
3.  RP = READ NP_APP, Reference_Port
4.  VERIFY RP <> 4194303

-- Check the rest of the chain
5.  WHILE (RP <> 4194303)
    {
6.      NPx = (Network Port, RP)
7.      VERIFY NPx, Protocol_Level <> (NON_BACNET_APPLICATION OR BACNET_APPLICATION)
    -- check that if PHYSICAL level it is the last NPO
8.      IF (NPx, Protocol_Level == PHYSICAL) THEN
9.          VERIFY NPx, Reference_Port = 4194303
10.     RP = READ NPx, Reference_Port
    }
    -- either the NPO is at PROTOCOL and does not reference another NPO or it is PHYSICAL

```

```

11. IF (Protocol_Revision >= 24 and NP_APP, Additional_Reference_Ports is present) THEN {
12.     REPEAT ARP = (for each entry in NP_APP, Additional_Reference_Ports) DO {
13.         NPx = (Network Port, ARP)
14.         VERIFY NPx, Protocol_Level <> (NON_BACNET_APPLICATION or BACNET_APPLICATION)
            -- check that if PHYSICAL level it is the last NPO
15.         IF (NPx, Protocol_Level == PHYSICAL) THEN
16.             IF (NPx, Additional_Reference_Ports is present) THEN
17.                 VERIFY NPx, Additional_Reference_Ports = (empty list)
            -- Check the rest of the chain
18.             RP = READ NPx, Reference_Port
19.             WHILE (RP <> 4194303)
            {
20.                 NPx = (Network Port, RP)
21.                 VERIFY NPx, Protocol_Level <> (NON_BACNET_APPLICATION
                    or BACNET_APPLICATION)
                -- check that if PHYSICAL level it is the last NPO
22.                 IF (NPx, Protocol_Level == PHYSICAL) THEN
23.                     VERIFY NPx, Reference_Port = 4194303
24.                     RP = READ NPx, Reference_Port
            }
        }
    }
}

```