BACnet® TESTING LABORATORIES ADDENDA

Addendum Fix to BTL Test Package 23.1

Revision final
Revised 12/5/2023

Approved by the BTL Working Group on March 30, 2023.
Approved by the BTL Working Group Voting Members on December 5, 2023;
Published on December 12, 2023.

[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-23.1 Fix-1: Testing of Current_Command_Priority property [BTLWG-1441] 2
BTL-23.1 Fix-2: 14.YY.2.1.2 Change destination virtual address specification [BTLWG-1309] 8
BTL-23.1 Fix-3: 14.YY.1.3.1 - Allow Certificate Authority and Hub to be Separate Devices [BTLWG-1461] 10
BTL-23.1 Fix-4: 14.YY.2.1.3 - Minimum NPDU Forwarding Calculation Size is Incorrect [BTLWG-1462] 11
BTL-23.1 Fix-5: 7.3.2.X62.5 - APDU_Length Test [BTLWG-1446] 13
BTL-23.1 Fix-6: 7.3.2.13.X3 - Reliability COMMUNICATION_FAILURE Test [BTLWG-1476] 15
BTL-23.1 Fix-7: Test Plan 9.9.1 - Data Link Layer - Secure Connect - Base Requirements Test [BTLWG-1474] 16
BTL-23.1 Fix-8: 9.9.1.3 - Silencing_Unsilencing Execution Test (CR-0563) [BTLWG-1475] 17

In the following document, language to be added to existing clauses within the BTL Test Package 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 Checklist, BTL Testplan, and BTL Specified Tests 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 shall 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-23.1 Fix-1: Testing of Current_Command_Priority property [BTLWG-1441]

Overview:

Current_Command_Priority property Tests from Interim TP 16.1_v8 were not included in any Test Packages. The proposed changes from Interim TP 16.1_v8 have been updated in this proposal.

Note: TP 16.1_v8 included a new test, 7.3.1.X1 Current_Command_Priority Tracking Test, that specifically tested the Current_Command_Priority property. The changes to test 7.3.1.3 below accomplish the same validation so the new test is not included in this update.

Changes:

Checklist Changes

None

Test Plan Changes

[Modify clauses 3.2.2, 3.3.3, 3.6.2, 3.7.5, 3.15.2, 3.16.4, 3.24.3, 3.25.3, 3.26.3, 3.27.3, 3.28.3, 3.29.3, 3.30.3, 3.31.3, 3.32.3, 3.33.3, 3.34.3, 3.35.3, 3.42.2, 3.54.2, 3.55.2]

3.?.? Supports Command Prioritization

The objects contain a priority array and support command prioritization.

BTL - 7.3.1.2 - Relinquish Default Test		
	Test Conditionality	If no object can be made to meet the configuration requirements, this test shall be skipped.
	Test Directives	
	Testing Hints	
BTL - 7.3.1.3 - Command Prioritization Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

[Modify Clause 11.2.2]

11.2.2 Supports Command Prioritization

Gateways are required to implement Priority_Array properties correctly with all 16 entries

BTL - 7.3.1.2 - Relinquish Default Test		
	Test Conditionality	Must be executed.
	Test Directives	The test shall be conducted upon an object which is representing information arriving through a Gateway. If no object can be made to meet the configuration requirements, this test shall be skipped.
	Testing Hints	
BTL - 7.3.1.3 - Command Prioritization Test		
	Test Conditionality	Must be executed.
	Test Directives	The test shall be conducted upon an object which is representing information arriving through a Gateway.
	Testing Hints	

[Modify Clause 3.6.7]

3.6.7 Supports Minimum_Off_Time

The object contains Minimum_Off_Time property.

BTL - 7.3.1.4 - Minimum_Off_Time		
	Test Conditionality	If the property is present, it must be executed.
	Test Directives	
	Testing Hints	
…		

[Modify Clause 3.6.8]

3.6.8 Supports Minimum_On_Time

The object contains Minimum_On_Time property.

BTL - 7.3.1.5 - Minimum_On_Time		
	Test Conditionality	If the property is present, it must be executed.
	Test Directives	
	Testing Hints	
…		

[Modify Clause 3.7.6]

3.7.6 Supports Minimum_Off_Time

The object contains Minimum_Off_Time property.

BTL - 7.3.1.4 - Minimum_Off_Time		
	Test Conditionality	If the property is present, it must be executed.
	Test Directives	
	Testing Hints	
…		

[Modify Clause 3.7.7]

3.7.7 Supports Minimum_On_Time

The object contains Minimum_On_Time property.

BTL - 7.3.1.5 - Minimum_On_Time		
	Test Conditionality	If the property is present, it must be executed.
	Test Directives	
	Testing Hints	
…		

Specified Test Changes

[Modify 135.1-2019 - 7.3.1.2]

7.3.1.2 Relinquish Default Test

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

~~BACnet Reference Clauses: 12.3.16, 12.4.12, 12.7.22, 12.8.20, 12.19.14, and 12.20.13.~~

Purpose: To verify that the Present_Value property takes on the value of Relinquish_Default when all prioritized commands have been relinquished. This test applies to *all*~~Analog Output, Analog Value, Binary Output, Binary Value, Multi-state Output, and Multi-state Value objects that are~~ commandable *objects*.

Test Concept: A pre-requisite to this test is that an object has been provided for which all prioritized commands have been relinquished and any minimum on/off time has been accounted for. The Present_Value is compared to the value of Relinquish_Default to ensure that they are the same. If possible, the value of Relinquish_Default is changed to verify that Present_Value tracks the changes.

Configuration Requirements: The object to be tested shall be configured such that all slots in the Priority_Array have a value of NULL and no internal algorithms are issuing prioritized commands to this object.

Test Steps:

1. VERIFY Priority_Array = (NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL)
2. *IF Protocol_Revision >= 17 THEN {*
   VERIFY Current_Command_Priority = NULL
   }
3. *READ X = Present_Value*
2. ~~TRANSMIT ReadProperty-Request,~~
   ~~'Object Identifier' = (the object being tested),~~
   ~~'Property Identifier' = Present_Value~~
3. ~~RECEIVE ReadProperty-ACK,~~
   ~~'Object Identifier' = (the object being tested),~~
   ~~'Property Identifier' = Present_Value~~
   ~~'Property Value' = (any valid value, X)~~
4. VERIFY Relinquish_Default = X
5. IF (Relinquish_Default is writable) THEN {
   WRITE Relinquish_Default = (any valid value, Y, other than *X*~~the one returned in step 3~~)
   VERIFY Present_Value = Y
   }

[Modify 135.1-2019 - 7.3.1.3]

7.3.1.3 Command Prioritization Test

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

~~BACnet Reference Clause: 19.2.~~

Purpose: To verify that the command prioritization algorithm is properly implemented. This test applies to all commandable objects.

Test Concept: The TD selects three different values V_{low}, V_{med}, and V_{high} chosen from the valid values specified in 4.4.2. For *objects that are limited to two values,* V_{low} and V_{high} shall be the same, and V_{med} shall be different. The TD also selects three priorities P_{low}, P_{med}, and P_{high}, all between 1 and 5, such that numerically $P_{low} > P_{med} > P_{high}$. The selected values are written one at a time to Present_Value at the corresponding priority. The Present_Value*, Current_Command_Priority* and Priority_Array are checked to verify correct operation. Priorities numerically smaller than 6 (higher priority) are used to eliminate minimum on/off time considerations.

Configuration Requirements: The object to be tested shall be configured such that all slots in the Priority_Array with a priority *numerically smaller* ~~higher~~ than 6 have a value of NULL.

Test Steps:

1. *VERIFY Priority_Array = (NULL, NULL, NULL, NULL, NULL, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)*

--

2. WRITE Present_Value = V_{low}, PRIORITY = P_{low}
3. VERIFY Present_Value = V_{low}
4. VERIFY Priority_Array = V_{low}, ARRAY INDEX = P_{low}

5. REPEAT Z = (each index 1 through 5 not equal to P_{low}) DO {
   VERIFY Priority_Array = NULL, ARRAY INDEX = Z
   }
6. *IF Protocol_Revision >= 17 THEN {*
   VERIFY Current_Command_Priority = P_{low}
   }

--

7. WRITE Present_Value = V_{high}, PRIORITY = P_{high}
8. VERIFY Present_Value = V_{high}
9. VERIFY Priority_Array = V_{high}, ARRAY INDEX = P_{high}
10. REPEAT Z = (each index 1 through 5 not equal to P_{low} or P_{high}) DO {
    VERIFY Priority_Array = NULL, ARRAY INDEX = Z
    }
11. *IF Protocol_Revision >= 17 THEN {*
    VERIFY Current_Command_Priority = P_{high}
    }

--

12. WRITE Present_Value = V_{med}, PRIORITY = P_{med}
13. VERIFY Present_Value = V_{high}
14. VERIFY Priority_Array = V_{med}, ARRAY INDEX = P_{med}
15. REPEAT Z = (each index 1 through 5 not equal to P_{low}, P_{med} or P_{high}) DO {
    VERIFY Priority_Array = NULL, ARRAY INDEX = Z
    }
16. *IF Protocol_Revision >= 17 THEN {*
    VERIFY Current_Command_Priority = P_{high}
    }

--

17. WRITE Present_Value = NULL, PRIORITY = P_{high}
18. VERIFY Present_Value = V_{med}
19. REPEAT Z = (each index 1 through 5 not equal to P_{low} or P_{med}) DO {
    VERIFY Priority_Array = NULL, ARRAY INDEX = Z
    }
20. *IF Protocol_Revision >= 17 THEN {*
    VERIFY Current_Command_Priority = P_{med}
    }

--

21. WRITE Present_Value = NULL, PRIORITY = P_{med}
22. VERIFY Present_Value = V_{low}
23. REPEAT Z = (each index 1 through 5 not equal to P_{low}) DO {
    VERIFY Priority_Array = NULL, ARRAY INDEX = Z
    }
24. *IF Protocol_Revision >= 17 THEN {*
    VERIFY Current_Command_Priority = P_{low}
    }

--

25. WRITE Present_Value = NULL, PRIORITY = P_{low}
27. *VERIFY Priority_Array = (NULL, NULL, NULL, NULL, NULL, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)*
~~28. REPEAT Z = (each index 1 through 5) DO {~~
~~VERIFY Priority_Array = NULL, ARRAY INDEX = Z~~
~~}~~
28. *IF Protocol_Revision >= 17 THEN {*
    VERIFY Current_Command_Priority is numerically larger than 5 or NULL
    }

[Modify 135.1-2019 - 7.3.1.4]

7.3.1.4 Minimum_Off_Time

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

~~BACnet Reference Clauses: 12.7.19, 12.8.17, 19.2.3, and Annex I.~~

Purpose: To verify that the minimum off time algorithm is properly implemented. ~~If minimum off time is not supported this test shall be omitted. This test applies to Binary Output and Binary Value objects.~~

Test Concept: The initial Present_Value of the object tested is set to ACTIVE and it is controlled at a priority numerically *larger*~~greater~~ (lower priority) than 6. The object has been in this state long enough for *the*~~any~~ minimum on time to have expired. The Present_Value is written to with a value of INACTIVE at priority 7. The value of slot 6 of the Priority_Array is monitored to verify that it contains the value INACTIVE for the duration of the minimum off time.

Configuration Requirements: The object to be tested shall be configured such that all slots in the Priority_Array numerically *smaller*~~less~~ than 7 have a value of NULL, the Present_Value is ACTIVE, and no internal algorithms are issuing commands to this object at a priority numerically *smaller*~~lower~~ (higher priority) that the priority that is currently controlling Present_Value.

Test Steps:

1. WRITE Present_Value = INACTIVE, PRIORITY = 7
2. VERIFY Present_Value = INACTIVE
3. VERIFY Priority_Array = INACTIVE, ARRAY INDEX = 6
4. *VERIFY Priority_Array = INACTIVE, ARRAY INDEX = 7*
5. *IF Protocol_Revision >= 17 THEN*
   VERIFY Current_Command_Priority = 6
6. WAIT (approximately 90% of Minimum_Off_Time ~~from step 1~~)
7. VERIFY Priority_Array = INACTIVE, ARRAY INDEX = 6
8. WAIT (**Minimum ON/OFF Fail Time** + Minimum_Off_Time ~~from step 1~~)
9. VERIFY Priority_Array = NULL, ARRAY INDEX = 6
10. *VERIFY Priority_Array = INACTIVE, ARRAY INDEX = 7*
11. *IF Protocol_Revision >= 17 THEN*
    VERIFY Current_Command_Priority = 7

[Modify 135.1-2019 - 7.3.1.5]

7.3.1.5 Minimum_On_Time

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

~~BACnet Reference Clauses: 12.7.20, 12.8.18, 19.2.3, and Annex I.~~

Purpose: To verify that the minimum on time algorithm is properly implemented. ~~If minimum on time is not supported this test shall be omitted. This test applies to Binary Output and Binary Value objects.~~

Test Concept: The initial Present_Value of the object tested is set to INACTIVE and it is controlled at a priority numerically *larger*~~greater~~ (lower priority) than 6. The object has been in this state long enough for *the*~~any~~ minimum *off*~~on~~ time to have expired. The Present_Value is written to with a value of ACTIVE at priority 7. The value of slot 6 of the Priority_Array is monitored to verify that it contains the value ACTIVE for the duration of the minimum on time.

Configuration Requirements: The object to be tested shall be configured such that all slots in the Priority_Array numerically *smaller*~~less~~ than 7 have a value of NULL, the Present_Value is INACTIVE, and no internal algorithms are issuing commands to this object at a priority numerically *smaller*~~lower~~ (higher priority) that the priority that is currently controlling Present_Value.

Test Steps:

1. WRITE Present_Value = ACTIVE, PRIORITY = 7
2. VERIFY Present_Value = ACTIVE
3. VERIFY Priority_Array = ACTIVE, ARRAY INDEX = 6
4. *VERIFY Priority_Array = ACTIVE, ARRAY INDEX = 7*
5. *IF Protocol_Revision >= 17 THEN*
   VERIFY Current_Command_Priority = 6

6~~4~~. WAIT (approximately 90% of Minimum_On_Time ~~from step 1~~)
7~~5~~. VERIFY Priority_Array = ACTIVE, ARRAY INDEX = 6
8~~6~~. WAIT (**Minimum ON/OFF Fail Time** + Minimum_On_Time ~~from step 1~~)
9~~7~~. VERIFY Priority_Array = NULL, ARRAY INDEX = 6
10. VERIFY Priority_Array = ACTIVE, ARRAY INDEX = 7
11. IF Protocol_Revision >= 17 THEN
VERIFY Current_Command_Priority = 7

BTL-23.1 Fix-2: 14.YY.2.1.2 Change destination virtual address specification [BTLWG-1309]

Overview:

14.YY.2.1.2 Change destination virtual address specification (CR-0532)

Changes:

Checklist Changes

None

Test Plan Changes

None

Specified Test Changes

[Modify 14.YY.2.1.2]

14.YY.2.1.2 Local Broadcast Execution Test

Reference: YY.5.3.3

Purpose: To verify that IUT, as a hub, correctly accepts and processes broadcast messages.

Test Concept: With the IUT operating as a hub, send a broadcast to the hub. Verify that the message is forwarded to all hub connectors except the one that originated it. Also verify that the hub's local node processes the broadcast.

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:

```
1.      TRANSMIT PORT (D4-IUT hub WebSocket),
            Encapsulated-NPDU,
            -- 'Originating Virtual Address' absent
            'Destination Virtual Address' =         (absent or X'FFFFFFFF', the local broadcast VMAC),
            -- 'Destination Options' absent
            'Data Options' =                        ({X'41'}), -- Secure Path
            'Payload'
                Who-Is-Request
2.      REPEAT Dx = (D2, D3) {
            RECEIVE PORT (Dx-IUT hub WebSocket),
                Encapsulated-NPDU,
                'Originating Virtual Address' =     (D4's VMAC),
                'Destination Virtual Address' =     X'FFFFFFFF',    -- the local broadcast VMAC
                -- 'Destination Options' absent
                'Data Options' =                    ({X'41'}), -- Secure Path
                'Payload'
```

Who-Is-Request
}
3. RECEIVE PORT (D4-IUT hub WebSocket),
Encapsulated-NPDU,
'Originating Virtual Address' = (IUT's VMAC)
'Destination Virtual Address' = (absent or X'FFFFFFFF', the local broadcast VMAC)
-- 'Destination Options' absent
'Data Options' = ({X'41'}), -- Secure Path
'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)

BTL-23.1 Fix-3: 14.YY.1.3.1 - Allow Certificate Authority and Hub to be Separate Devices [BTLWG-1461]

Overview:

14.YY.1.3.1 specifies that the TD is the CA and the Hub. CR-0558 allows the CA and Hub to be separate devices. CR-0558 also allows the Issuer Certificate to be added to the IUT.

Changes:

Checklist Changes

None

Test Plan Changes

None

Specified Test Changes

[Modify 14.YY.1.3.1]

14.YY.1.3.1 Configuration Via PEM Test

Reference: YY.7.4.1.3

Purpose: To verify that the IUT's configuration tool supports PEM format certificates.

Test Concept: The IUT's configuration tool is made to export a certificate signing request in PEM format. The PEM signing request is imported into the Certificate Authority (CA) and ~~a PEM format~~ an operational certificate is exported in PEM format along with the PEM formatted issuer certificate. The operational certificate is then loaded into the IUT with the IUT's configuration tool. If the IUT requires the issuer certificate it shall also be loaded into the IUT with the IUT's configuration tool. The IUT is then configured to connect to the TD as the primary hub. The IUT is allowed to connect to the primary hub, and a successful connection is verified.

Test Steps:

1. MAKE(the IUT's configuration tool export a certificate signing request in PEM format)
2. CHECK(that the PEM file is well formed)
3. MAKE(import the PEM file into the CA~~TD~~ and generate a PEM formatted operational certificate)
4. MAKE(the CA generate a PEM formatted issuer certificate)
5. MAKE(the IUT's configuration tool load the operational certificate into the IUT)
6. IF (the IUT requires the issuer certifcate) THEN
   MAKE(the IUT's configuration tool load the issuer certificate into the IUT)
7. MAKE(the IUT connect to the TD using the new certificate)

BTL-23.1 Fix-4: 14.YY.2.1.3 - Minimum NPDU Forwarding Calculation Size is Incorrect [BTLWG-1462]

Overview:

The calculation of the minimum NPDU forwarding size is incorrect. See CR-0559.

The Data Options field contains two Header Options.

Header Option #1
[1 byte] X'C1' - another Header Options follows, Must be understood, Header Length and Header Data are absent, Option Type is Secure Path.

Header Option #2
[1 byte] X'3F' - last Header Option, Header Option can be ignored, Header Length and Header Data are present, Option Type is Proprietary.
[2 bytes] Header Length. (X'015C')
[4188 bytes] Header Data
    [2 bytes] Vendor Identifier (X'0000')
    [1 byte] Proprietary Option Type (X'34')
    [4185 bytes] Proprietary Header Data

Changes:

Checklist Changes

None

Test Plan Changes

None

Specified Test Changes

[Modify 14.YY.2.1.3]

14.YY.2.1.3 Minimum NPDU Forwarding Size Test

Reference: YY.5.1

Purpose: To verify that the hub can forward BVLC messages of length 1497 with 4192 octets of data and destination options.

Test Concept: With the IUT operating as the primary hub, connect devices D3 and D4 to the IUT. D3 sends a BVLC of length 1497 octets with 4192 octets of data options to D4 via the hub. Verify that the BVLC is correctly forwarded to D4.

Configuration Requirements: The IUT is configured as a primary or failover hub and the test devices D3, D4 and D5 are connected to it.

Test Steps:

1. TRANSMIT PORT (D3-IUT hub WebSocket),
    Encapsulated-NPDU,
    -- 'Originating Virtual Address' absent
    'Destination Virtual Address' = (D4's VMAC),
    -- 'Destination Options' absent

'Data Options' = ({X'C1',
X'3F
X'105C - Header Data length = 4188 octets
2 octets - V1, any vendor identifier <> IUT Vendor Identifier
1 octet - OT1, any proprietary option type
4185 octets - Payload, any value
}), ~~-- replace … with 4186 octets of any value 'Payload'~~
WriteProperty-Request,
'Object Identifier' = (O: any object identifier),
'Property Identifier' = (P: any property identifier),
'Property Value' = (V: any data value with an encoded length which makes the PayLoad 1497 octets)

2. RECEIVE PORT (D4-IUT hub WebSocket),
Encapsulated-NPDU,
'Originating Virtual Address' = (D3's VMAC),
-- 'Destination Virtual Address' absent
-- 'Destination Options' absent
'Data Options' = ({X'C1',
X'3F
X'105C - Header Data length = 4188 octets
2 octets - V1
1 octet - OT1
4185 octets - Payload
}),
WriteProperty-Request,
'Object-Identifier' = O,
'Property-Identifier' = P,
'Property Value' = V

BTL-23.1 Fix-5: 7.3.2.X62.5 - APDU_Length Test [BTLWG-1446]

Overview:

Erratum.

In Step 2, third IF, should be '<' instead of '<>'.

…

IF MAX_APDU <> NP.APDU_Length THEN

MAX_APDU = NP.APDU_Length

Changes:

Checklist Changes

None

Test Plan Changes

None

Specified Test Changes

[Modify 7.3.2.X62.5]

7.3.2.X62.5 APDU_Length Test

Reason for Change: New test per Addendum 135-2012ai.

Purpose: To verify that the Device object does not report a Max_APDU_Length_Accepted that is larger than the largest value reported by the configured and enabled Network Port objects.

Test Concept: Determine the largest APDU_Length property for all configured and enabled Network Port objects with a Protocol_Level of BACNET_APPLICATION. Verify that each is larger than 50 and less than or equal the maximum allowed for the attached datalink. Verify that the Max_APDU_Length_Supported property of the Device object is not larger than that maximum.

1. MAX_APDU = 0
2. REPEAT NP = (all configured and enabled Network Port objects with a
   Protocol_Level of BACNET_APPLICATION) {
   IF NP.APDU_Length < 50 THEN
   ERROR "APDU_Length must not be less than 50."
   IF NP.APDU_Length > (the maximum allowable for the Network_Type) THEN
   ERROR "APDU_Length is too large for the connected Network_Type"
   IF MAX_APDU < NP.APDU_Length THEN
   MAX_APDU = NP.APDU_Length
   }
3. VERIFY (Device, 4194303), Max_APDU_Length_Supported <= MAX_APDU

Note to Tester: the maximum allowable APDU_Length for a network type should be calculated from the maximum NPDU size minus 21 according to SSPC interpretation IC135-2020-2.

BTL-23.1 Fix-6: 7.3.2.13.X3 - Reliability COMMUNICATION_FAILURE Test [BTLWG-1476]

Overview:

Erratum.

Remove Step 3 as Step 4 does the same thing.

Changes:

Checklist Changes

None

Test Plan Changes

None

Specified Test Changes

[Modify 7.3.2.13.X3]

7.3.2.13.X3 Reliability COMMUNICATION_FAILURE Test

Reason for Change: New Tests for Global Group object type. COMMUNICATION_FAILURE does not include errors returned by an object.

Purpose: This test case verifies that the Member_Status_Flags FAULT flag will remain FALSE while the Reliability property is COMMUNICATION_FAILURE.

Test Concept: Force a member of the Group_Members property to stop communicating and verify the Reliability property equals COMMUNICATION_FAILURE and the Member_Status_Flags FAULT flag remains FALSE.

Configuration Requirements: The IUT shall be configured with a Global Group object with the Group_Members containing a member M1 at index N1that can be made to discontinue communications. The Out_Of_Service property of the Global Group object must remain FALSE throughout the test. W1 is the maximum time it takes for the Global Group to receive an update from M1.

Notes to Tester: Reliability will change to COMMUNICATION_FAILURE when a member is no longer able to communicate its Status_Flags property. This can occur when the device goes offline.

Test Steps:

1. MAKE (M1 discontinue communications)
2. WAIT (W1)
4. IF (Reliability is present) THEN
   VERIFY Reliability = COMMUNICATION_FAILURE
5. VERIFY Member_Status_Flags = {?, FALSE, ?, ?}

BTL-23.1 Fix-7: Test Plan 9.9.1 - Data Link Layer - Secure Connect - Base Requirements Test [BTLWG-1474]

Overview:

Several B/SC nodes have been identified that do not support a URI with a valid path as defined in RFC 6455.

Changes:

Checklist Changes

None

Test Plan Changes

9.9.1 Base Requirements

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

BTL - 14.YY.1.1.1 - Connect and Maintain Hub Connection Test		
	Test Conditionality	Must be executed.
	Test Directives	Repeat with IUT configured with a hub URI that requires DNS resolution, with a URI that does not, and with a URI that contains a valid path.
	Testing Hints	
…		

Specified Test Changes

None

BTL-23.1 Fix-8: 9.9.1.3 - Silencing_Unsilencing Execution Test (CR-0563) [BTLWG-1475]

Overview:

The test is overly restrictive on what a visual and audible appliance is.

Changes:

Checklist Changes

None

Test Plan Changes

None

Specified Test Changes

9.9.1.3 Silencing/Unsilencing Execution Tests
~~**9.9.3 Silencing/Unsilencing Execution Tests**~~

~~BACnet Reference Clause: 13.13.~~

Purpose: To verify that the IUT can correctly execute a LifeSafetyOperation service request to silence *and unsilence* an alarming device.

Test Concept: ~~An audible device and/or visual device is attached to the IUT and is sounding/flashing,~~ *A life safety object, O1, enters a non-normal state and an audible and/or visual indication, I1, occurs.* A LifeSafetyOperation service request is transmitted to silence *I1 and I1 is verified to be silenced. A second LifeSafetyOperation service request is transmitted to unsilence I1 and I1 is verified to be unsilenced. The Silenced property is also validated.* ~~the sounder/strobe.~~

There are different allowable BACnetSilencedState values based on the silence operation performed and the setup of the IUT. In the below tables, N/A marks an operation that is inappropriate for the test with the corresponding IUT setup.

Audible Only Indication			
Silence Request (S)	***Allowable Silenced State (S_State)***	***Unsilenced Request (U)***	***Allowable Silenced State (U_STATE)***
SILENCE	*ALL_SILENCED, AUDIBLE_SILENCED*	*UNSILENCE*	*UNSILENCED, proprietary*
SILENCE_AUDIBLE	*ALL_SILENCED, AUDIBLE_SILENCED*	*UNSILENCE_AUDIBLE*	*UNSILENCED, proprietary*
SILENCE_VISUAL	*N/A*	*UNSILENCE_VISUAL*	*N/A*

Visual Only Indication			
Silence Request (S)	***Allowable Silenced State (S_State)***	***Unsilenced Request (U)***	***Allowable Silenced State (U_State)***
SILENCE	*ALL_SILENCED, VISUAL_SILENCED*	*UNSILENCE*	*UNSILENCED, proprietary*

SILENCE_AUDIBLE	*N/A*		*UNSILENCE_AUDIBLE*	*N/A*
SILENCE_VISUAL	*ALL_SILENCED, VISUAL_SILENCED*		*UNSILENCE_VISUAL*	*UNSILENCED, proprietary*

Audible and Visual Indication				
Silence Request (S)	***Allowable Silenced State (S_State)***		***Unsilenced Request (U)***	***Allowable Silenced State (U_State)***
SILENCE	*ALL_SILENCED*		*UNSILENCE*	*UNSILENCED, proprietary*
SILENCE_AUDIBLE	*AUDIBLE_SILENCED*		*UNSILENCE_AUDIBLE (all silenced)*	*SILENCED_VISUAL, proprietary*
			UNSILENCE_AUDIBLE (audible silenced, visual active)	*UNSILENCED, proprietary*
			UNSILENCE_AUDIBLE (audible active, visual silenced)	*N/A*
SILENCE_VISUAL	*VISUAL_SILENCED*		*UNSILENCE_VISUAL (all silenced)*	*SILENCED_AUDIBLE, proprietary*
			UNSILENCE_VISUAL (audible silenced, visual active)	*N/A*
			UNSILENCE_VISUAL (audible active, visual silenced)	*UNSILENCED, proprietary*

Configuration Requirements: The IUT must *support an indication that audible and/or visual equipment has been the silenced and unsilenced.* ~~be fitted with needed audible and visual equipment.~~

Note to Tester: The indication of silencing and unsilencing an audible and visual change is vendor specific but must be detectable by the lab during testing.

Test Steps:

REPEAT S = (for each supported LifeSafetyOperation service Request specified in the above table) DO {
1. *MAKE (O1 enter a condition, that S can silence)*
2. *VERIFY Silenced = (UNSILENCED or a proprietary value with a similar semantic)*
3. *TRANSMIT LifeSafetyOperation-Request,*
   'Requesting Process Identifier' = (any valid identifier),
   'Requesting Source' = (any valid character string),
   'Request' = S,
   'Object Identifier' = (O1 or absent)
4. *RECEIVE BACnet-SimpleACK-PDU*
5. *CHECK (I1 is silenced)*
6. *VERIFY Silenced = (S_State)*
7. *TRANSMIT LifeSafetyOperation-Request,*
   'Requesting Process Identifier' = (any valid identifier),
   'Requesting Source' = (any valid character string),
   'Request' = U,
   'Object Identifier' = (O1 or absent)
8. *RECEIVE BACnet-SimpleACK-PDU*
9. *CHECK (I1 is unsilenced)*
10. *VERIFY Silenced = (U_State)*

}

1. REPEAT X = (All supported enumerations that silence the object) DO {
2. MAKE (the selected object enter a state where enumeration X will commence alerts)
3. MAKE (Event_State = NORMAL)
4. TRANSMIT LifeSafetyOperation Request,
'Requesting Process Identifier' = (any valid identifier),
'Requesting Source' = (any valid character string),
'Request' = (any valid LifeSafetyOperation request),
'Object Identifier' = (the selected object)
5. RECEIVE BACnet SimpleACK PDU
6. CHECK (that the sounder/strobe is inactive)
}