



BACnet[®] TESTING LABORATORIES ADDENDA

Addendum imp3 to BTL Test Package 23.3

**Revision final
Revised 10/21/2024**

Approved by the BTL Working Group on October 3, 2024;
Approved by the BTL Working Group Voting Members on October 17, 2024;
Published on October 22, 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 imp3-1: Test Concept Clarification for Ensuring 5 Concurrent COV Subscribers [BTLWG-1432]	2
BTL-23.3 imp3-2: 7.3.1.16 Array Resizing Test [BTLWG-1464]	4
BTL-23.3 imp3-3: Fix Broadcast Distribution via Hostname Test [BTLWG-1479]	6
BTL-23.3 imp3-4: Update Active_COV_Subscriptions SubscribeCOV Test [BTLWG-1483]	8
BTL-23.3 imp3-5: Verify Tframe_gap Test [BTLWG-1486]	11
BTL-23.3 imp3-6: Already Running Timer Restarted with Default_Timeout [BTLWG-1487]	12
BTL-23.3 imp3-7: Resizing Group_Member_Names Test [BTLWG-1488]	14
BTL-23.3 imp3-8: WritePropertyMultiple Allowed Errors [BTLWG-1491]	17
BTL-23.3 imp3-9: Exclude Elevator Objects for DS-AM-A [BTLWG-1511]	19
BTL-23.3 imp3-10: Change Test Conditionality for Sections 4.2.1 and 4.4.1 in Test Plan [BTLWG-1516]	20
BTL-23.3 imp3-11: Clarify DS-WP-B and DS-WPM-B Test Directives [BTLWG-1526]	23
BTL-23.3 imp3-12: AE-N-E-B Removal of Duplicate Entry [BTLWG-512]	31
BTL-23.3 imp3-13: Make Intrinsic Reporting Consistent [BTLWG-1312]	32
BTL-23.3 imp3-14: Add COV A and B Support for Other Standard Object Types [BTLWG-1373]	43
BTL-23.3 imp3-15: Timestamp format for test 9.1.2.1 [BTLWG-1524]	47
BTL-23.3 imp3-16: Add DS-COV-B Tests for the Load Control Object [BTLWG-1544]	50
BTL-23.3 imp3-17: Fix Test Plan Descriptions in DS-COV-B [BTLWG-1545]	52
BTL-23.3 imp3-18: DM-OCD-A Change to Allow any Version Client to Create New Objects [BTLWG-636]	53
BTL-23.3 imp3-19: DM-R-B and DS-COVU-B Clarification [BTLWG-1559]	55
BTL-23.3 imp3-20: B_SC Must Support Time Synchronization [BTLWG-1564]	57
BTL-23.3 imp3-21: Do Not Allow Absent DVA for Broadcast [BTLWG-1566]	59
BTL-23.3 imp3-22: Register Foreign Device Tests when NPO Supported [BTLWG-1444]	61

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 ~~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-23.3 imp3-1: Test Concept Clarification for Ensuring 5 Concurrent COV Subscribers [BTLWG-1432]**Overview:**

The Test Concept does not clearly describe the actual test, confusing the reader.

Changes:

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, change all occurrences of test 9.10.1.11 from 135.1-2023 to BTL]

Specified Test Changes

9.10.1.11 Ensuring 5 Concurrent COV Subscribers

Reason For Change: Clarified the Test Concept and added Test Conditionality.

Purpose: This test case verifies that the IUT can support 5 concurrent subscriptions.

Test Concept: Have the TD subscribe with 5 different process identifiers, V_1 through V_5 for monitored object $O1$. After each subscription verify a corresponding notification is sent. Change the monitored object and verify that all 5 notifications were sent, and then check to ensure that 5 notifications are sent when the monitored object changes.

Test Conditionality: The IUT should not have any subscriptions at the start of this test.

Notes to Tester: The notification in step 3 can be received in any order by the TD.

Test Steps

```

1. REPEAT (X= $V_1$  to  $V_5$ ) DO {
    TRANSMIT SubscribeCOV-Request,
        'Subscriber Process Identifier' = X,
        'Monitored Object Identifier' = O1 (any object supporting COV notifications),
        'Issue Confirmed Notifications' = TRUE | FALSE,
        'Lifetime' = (any valid value that will allow the subscription to outlast the test)
    RECEIVE BACnet-SimpleACK-PDU
    IF (if confirmed notifications were requested) THEN
        BEFORE Notification Fail Time
        RECEIVE ConfirmedCOVNotification-Request,
            'Subscriber Process Identifier' = X,
            'Initiating Device Identifier' = IUT,
            'Monitored Object Identifier' = O1 (the same object used in the subscription),
            'Time Remaining' = (any valid value),
            'List of Values' = (the initial Present_Value and initial Status_Flags)
        TRANSMIT BACnet-SimpleACK-PDU
    ELSE
        BEFORE Notification Fail Time
        RECEIVE UnconfirmedCOVNotification-Request,
            'Subscriber Process Identifier' = X,
            'Initiating Device Identifier' = IUT,
            'Monitored Object Identifier' = O1 (the same object used in the subscription),
            'Time Remaining' = (any valid value),

```

```

        'List of Values' =                (the initial Present_Value and initial Status_Flags)
    }
2.  MAKE (Present_Value = any value that differs from "initial Present_Value" such that a COV notification would be
generated)
3.  REPEAT (X=V1 to V5) DO {
    IF (#confirmed notifications were requested) THEN
        RECEIVE ConfirmedCOVNotification-Request,
            'Subscriber Process Identifier' =    X,
            'Initiating Device Identifier' =    IUT,
            'Monitored Object Identifier' =    OI(the same object used in the subscription),
            'Time Remaining' =                (any valid value),
            'List of Values' =                (the new Present_Value and Status_Flags)
        TRANSMIT BACnet-SimpleACK-PDU
    ELSE
        RECEIVE UnconfirmedCOVNotification-Request,
            'Subscriber Process Identifier' =    X,
            'Initiating Device Identifier' =    IUT,
            'Monitored Object Identifier' =    OI(the same object used in the subscription),
            'Time Remaining' =                (any valid value),
            'List of Values' =                (the new Present_Value and Status_Flags)
    }

```

BTL-23.3 imp3-2: 7.3.1.16 Array Resizing Test [BTLWG-1464]

Overview:

In this test, which is located in the document ANSI/ASHRAE Standard 135.1-2019/ 135.1-2023, values N1 ... N6 are defined, with estimates 'greater than or equal to'(\geq) and 'less than or equal to'(\leq). These conditions contradict the testing concept that the array should be decreased and increased. As the test is currently written, it could be conducted without ever altering the size of the array.

Changes:

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, change reference to test 7.3.1.16 to use the name used in 135.1-2023]

4.6.4 Contains Resizable Array Properties

The IUT contains, or can be made to contain, an array property that is resizable by writing to the 0th element.

BTL - 9.22.1.X1 - Writing an Array Size		
	Test Conditionality	This test shall be executed on a single instance of each resizable property, both standard and proprietary, that do not have specific tests for those properties.
	Test Directives	
	Testing Hints	
BTL - 7.3.1.16 - Array Sizing Test Array Resizing Test		
	Test Conditionality	This test shall be executed if the IUT is protocol revision 4 or higher on a single instance of each resizable property, both standard and proprietary, that do not have specific tests for those properties.
	Test Directives	
	Testing Hints	

Specified Test Changes

[Move copy of 135.1 test into BTL Specified Tests and change as shown below.]

7.3.1.16 Array Resizing Test

Reason for Change: Modified the test steps to match the Test Concept of < and > vs <= and >=.

The test in this clause shall be applied to resizable arrays in devices claiming Protocol_Revision 4 or higher. It may be applied to resizable arrays in devices claiming Protocol_Revision 3 or lower, but only where conformance to the rules on resizing arrays of Protocol_Revision 4 is claimed.

Purpose: To verify that resizable arrays are resized in accordance with the rules added in Protocol_Revision 4.

Test Concept: The array is written as a whole to set it to a non-zero size. It is then resized smaller and larger by writing the entire array. It is then resized smaller and larger by writing to element number zero. An attempt is made to increase it with an invalid write. After each operation, the array size and array contents are checked. Finally, if it can be resized to have zero elements, it is then written to size zero. If possible, all elements in the arrays should be distinguishable from each other and across write operations.

Test Steps:

1. WRITE (the array property being tested) = (array of non-zero size N1)
2. VERIFY (array is as written in step 1)
3. WRITE (the array property being tested) = (array of non-zero size N2, ~~where $N2 \leq N1$~~ where $N2 < N1$) 4. VERIFY (array is as written in step 3)
5. WRITE (the array property being tested) = (array of non-zero size N3, ~~where $N3 \geq N1$~~ where $N3 > N1$)
6. VERIFY (array is as written in step 5)
7. WRITE (the array property being tested) = (a non-zero unsigned value N4, ~~where $N4 \leq N1$~~ where $N4 < N1$), ARRAY INDEX = 0
8. VERIFY (array contains first N4 elements of the array written in step 5)
9. WRITE (the array property being tested) = (N5, ~~where $N5 \geq N4$~~ where $N5 > N4$), ARRAY INDEX = 0
10. VERIFY (array contains first N4 elements of the array written in step 5, plus N5 – N4 additional elements, initialized to particular values if specified for the array property being tested)
11. TRANSMIT WriteProperty-Request,
 - 'Object Identifier' = (the object being tested),
 - 'Property Identifier' = (the array property being tested),
 - 'Property Array Index' = (N6, ~~where $N6 \geq N5$~~ where $N6 > N5$),
 - 'Property Value' = (one array element)
12. RECEIVE BACnet-Error-PDU
 - Error Class = PROPERTY,
 - Error Code = INVALID_ARRAY_INDEX
13. VERIFY (array is unchanged from step 10)
14. IF (the array can be resized to have zero elements) THEN
 - WRITE (the array property being tested) = (empty array)
 - VERIFY (array is empty)

BTL-23.3 imp3-3: Fix Broadcast Distribution via Hostname Test [BTLWG-1479]**Overview:**

An error in test 12.3.11.4 (14.10X.4) was found in that a TRANSMIT step was missing between steps 8 and 9. Further review of the test revealed some other areas of improvement.

Changes:

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, modify all references of 135.1-2023 - 12.3.11.4 to BTL]

Specified Test Changes

[From 135.1-2023 move test into BTL Specified Tests, modify as shown]

12.3.11.4 Broadcast Distribution Table Configuration via Hostname Entries

Reason for Change: With the advent of Network Port objects, BBMDs now need to accept hostname BDT entries.

Purpose: Verify that the IUT accepts and resolves hostname entries in the BBMD_Broadcast_Distribution_Table *and that the resolved IP address are shown in the result of a Read-Broadcast-Distribution-Table request.*

Test Concept: Fill the BBMD_Broadcast_Distribution_Table with 4 entries: the IUT, an entry with an IP address (IP1), an entry with a resolvable hostname, *HN1 that resolves to an IP address, IP2, (at IP address IP2),* and an entry with a non-resolvable hostname (*HN2*). Send a broadcast that the IUT should distribute to its peer BBMDs and verify that it sends *them* to the resolvable entries. Verify that the Broadcast Distribution Table contains the correct entries.

Configuration Requirements: The IUT is configured to operate as a BBMD and the TD (*D1*) is located on the same IP subnet.

Notes to Tester: The Forwarded-NPDU messages can be received in any order.

Test Steps:

1. WRITE BBMD_Broadcast_Distribution_Table =(4 entries:
 - the IUT,
 - IP1 (an entry with an IP address),*
 - HN1 (an entry with a resolvable hostname),*
 - HN2 (an entry with a non-resolvable hostname))*
2. *READ BDT = BBMD Broadcast Distribution Table*
3. *CHECK (BDT contains IUT, IP1, HN1, HN2 in any order)*
34. TRANSMIT ReinitializeDevice-Request
 - 'Reinitialized State of Device' = ACTIVATE_CHANGES
35. WAIT **Activate Changes Fail Time**
46. WAIT until the IUT completes DNS resolution
57. TRANSMIT
 - DA = Local IP Broadcast,
 - SA = D1,
 - Original-Broadcast-NPDU,
 - NPDU = Who-Is-Request
68. RECEIVE
 - DA = IP1,
 - SA = IUT,
 - Forwarded-NPDU,

Originating-Device = D1,
NPDU = Who-Is

79. RECEIVE

DA = IP2,
SA = IUT,
Forwarded-NPDU,
Originating-Device = D1,
NPDU = Who-Is

810. READ BDT = BBMD_Broadcast_Distribution_Table — re-read the table to determine the order the IUT placed the entries

11. CHECK (BDT contains IUT, IP1, HN1, HN2 in any order)

12. TRANSMIT

DA = IUT,
SA = D1,
Read-Broadcast-Distribution-Table

4013. RECEIVE Read-Broadcast-Distribution-Table-Ack,
'List of BDT Entries' = (4 entries:
the IUT's IP address,
the IP address entry,
the IP address for the resolved hostname entry,
X'000000000000' for the non-resolvable entry,
in the same order as BDT) read from
BBMD_Broadcast_Distribution_Table)

BTL-23.3 imp3-4: Update Active_COV_Subscriptions SubscribeCOV Test [BTLWG-1483]

Overview:

The Test Concept has confusing language.

Clause 12.11. 39 states: “If the subscribed-to property represents a numeric quantity, the COV Increment in use for the COV subscription shall be included in the Active_COV_Subscriptions entry otherwise it is a local matter whether the COV Increment is included in the Active_COV_Subscriptions entry. “

Changes:

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, modify all references to test 7.3.2.10.1 from 135.1-2023 to BTL]

Specified Test Changes

7.3.2.10.1 Active_COV_Subscriptions SubscribeCOV Test

Reason for Change: Double negation was confusing in the Test Concept.

Purpose: This test case verifies that the IUT correctly updates the Active_COV_Subscriptions property when COV subscriptions are created, cancelled and timed-out using SubscribeCOV.

Test Concept: INC₁, INC₂, and INC₃ are each **not** present if the property is **not** numeric; **present if a valid Increment was provided in the subscription**; and optionally present otherwise.

Configuration Requirements: In this test, the tester shall choose three standard objects, O₁, O₂, and O₃, for which the device supports SubscribeCOV. O₁, O₂, and O₃ are not required to refer to different objects. The tester shall also choose three nonzero unique process identifiers, P₁, P₂, and P₃, and three non-zero lifetimes L₁, L₂, and L₃. Lifetime L₁ shall be long enough to allow the initial part of the test to run through to step 14. Lifetimes L₂ and L₃ shall be long enough for the whole test to be completed without expiring.

The IUT shall start the test with no entries in its Active_COV_Subscriptions property.

Test Steps:

1. TRANSMIT SubscribeCOV-Request,
 - 'Subscriber Process Identifier' = P₁,
 - 'Monitored Object Identifier' = O₁,
 - 'Issue Confirmed Notifications' = TRUE,
 - 'Lifetime' = L₁
2. RECEIVE BACnet-SimpleACK-PDU
3. BEFORE **Notification Fail Time**
 - RECEIVE ConfirmedCOVNotification-Request,
 - 'Subscriber Process Identifier' = P₁,
 - 'Initiating Device Identifier' = IUT,
 - 'Monitored Object Identifier' = O₁,
 - 'Time Remaining' = (a value approximately equal to L₁),
 - 'List of Values' = (values appropriate to the object type of the monitored object)

4. TRANSMIT BACnet-SimpleACK-PDU
5. IF P1 is numeric THEN
 - VERIFY Active_COV_Subscriptions = { { {TD, P1}, {O1, Present_Value }, TRUE, (a value less than L1), (INC1 : not present or a valid Increment) } }
- ELSE
 - VERIFY Active_COV_Subscriptions = { { {TD, P1, { O1, Present_Value }, TRUE, (a value less than L1), (INC1: not present) } }
6. TRANSMIT SubscribeCOV-Request,
 - 'Subscriber Process Identifier' = P2,
 - 'Monitored Object Identifier' = O2,
 - 'Issue Confirmed Notifications' = FALSE,
 - 'Lifetime' = L2
7. RECEIVE BACnet-SimpleACK-PDU
8. BEFORE **Notification Fail Time**
 - RECEIVE UnconfirmedCOVNotification-Request,
 - 'Subscriber Process Identifier' = P2,
 - 'Initiating Device Identifier' = IUT,
 - 'Monitored Object Identifier' = O2,
 - 'Time Remaining' = (a value approximately equal to L2),
 - 'List of Values' = (values appropriate to the object type of the monitored object)
9. VERIFY Active_COV_Subscriptions = { { {TD, P1}, {O1, Present_Value}, TRUE, (a value less than L1), INC1}, { {TD, P2}, {O2, Present_Value}, FALSE, (a value less than L2), (INC2: not present if the property is not numeric; present if a valid Increment was provided in the subscription; optionally present otherwise) } }
10. TRANSMIT SubscribeCOV-Request,
 - 'Subscriber Process Identifier' = P3,
 - 'Monitored Object Identifier' = O3,
 - 'Issue Confirmed Notifications' = FALSE,
 - 'Lifetime' = L3
11. RECEIVE BACnet-SimpleACK-PDU
12. BEFORE **Notification Fail Time**
 - RECEIVE UnconfirmedCOVNotification-Request,
 - 'Subscriber Process Identifier' = P3,
 - 'Initiating Device Identifier' = IUT,
 - 'Monitored Object Identifier' = O3,
 - 'Time Remaining' = (a value approximately equal to L3),
 - 'List of Values' = (values appropriate to the object type of the monitored object)
13. IF P3 is numeric THEN
 - VERIFY Active_COV_Subscriptions = { { {TD, P1}, {O1, Present_Value}, TRUE, (a value less than L1), INC1}, { {TD, P2}, {O2, Present_Value}, FALSE, (a value less than L2), INC2}, { {TD, P3}, {O3, Present_Value}, FALSE, (a value less than L3), INC3: not present or (a valid Increment) } }
- ELSE
 - VERIFY Active_COV_Subscriptions = { { {TD, P1}, {O1, Present_Value}, TRUE, (a value less than L1), INC1 }, { {TD, P2}, {O2, Present_Value}, FALSE, (a value less than L2), INC2 }, { {TD, P3}, {O3, Present_Value}, FALSE, (a value less than L3), (INC3: not present) } }
14. WAIT L1 + the IUT's timer granularity
15. VERIFY Active_COV_Subscriptions = { {TD, P 2 }, {O 2 , Present_Value}, FALSE, (a value less than L 2), INC2 (a valid Increment if the property is REAL)}, { {TD, P 3 }, {O 3 , Present_Value}, FALSE, (a value less than L 3), INC3(a valid Increment if the property is REAL) } }
16. TRANSMIT SubscribeCOV-Request,
 - 'Subscriber Process Identifier' = P3 ,
 - 'Monitored Object Identifier' = O3
17. RECEIVE BACnet-SimpleACK-PDU
18. VERIFY Active_COV_Subscriptions = { { {TD, P 2 }, {O 2 , Present_Value}, FALSE, (a value less than L 2), INC2(a valid Increment if the property is REAL) } }
19. TRANSMIT SubscribeCOV-Request,
 - 'Subscriber Process Identifier' = P2 ,
 - 'Monitored Object Identifier' = O2

20. RECEIVE BACnet-SimpleACK-PDU
21. VERIFY Active_COV_Subscriptions = { }

BTL-23.3 imp3-5: Verify Tframe_gap Test [BTLWG-1486]

Overview:

A token and poll-for-master is only executed by master devices. In this respect, this sentence “Test both maintenance (Token and Poll_For_Master) as well as data frames.” should be inserted in the master test and removed in the slave test.

Changes:

Checklist Changes

None

Test Plan Changes

9.1 Data Link Layer - MS/TP - Master Node

9.1.1 Base Requirements

135.1-2023 - 12.1.3.3 - Verify Tframe_gap		
	Test Method	Manual
	Test Conditionality	Must be executed.
	Test Directives	Every MS/TP device shall claim 9600 baud. Test that the device operates at each baud rate that is claimed. Devices claiming Protocol_Revision 12 or higher where Addendum 135-2008ab is incorporated, shall claim 38400.. <i>Test both maintenance (Token and Poll_For_Master) as well as data frames.</i>
	Testing Hints	

9.2 Data Link Layer - MS/TP - Slave Node

9.2.1 Base Requirements

135.1-2023 - 12.1.3.3 - Verify Tframe_gap		
	Test Method	Manual
	Test Conditionality	Must be executed.
	Test Directives	Every MS/TP device shall claim 9600 baud. Test that the device operates at each baud rate that is claimed. Devices claiming Protocol_Revision 12 or higher where Addendum 135-2008ab is incorporated, shall claim 38400.. <i>Test both maintenance (Token and Poll_For_Master) as well as data frames.</i>
	Testing Hints	

Specified Test Changes

None

BTL-23.3 imp3-6: Already Running Timer Restarted with Default_Timeout [BTLWG-1487]

Overview:

The properties "Initial_Value" and "Default_Value" are in "Test Concept" and "Configuration Requirements". This does not exist according to standard 135-2020.

The correct ones would be: "Initial_Timeout" and "Default_Timeout"

Changes:

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, change all references of 135.1-2023 - 7.3.2.47.1.7 to BTL]

[Section 3.57.6 also needs the Test Conditionality Changed.]

3.57.6 Supports Default_Timeout

135.1-2023 - 7.3.2.47.1.7 - Already Running Timer Restarted with Default_Timeout		
	Test Conditionality	If the IUT does not support a writable Timer_Running in any Timer object which contains a Default_Timeout, this test shall be skipped. If every Timer only goes into RUNNING state with an Initial_ValueInitial_Timeout equal to Default_ValueDefault_Timeout, this test shall be skipped.
	Test Directives	
	Testing Hints	

Specified Test Changes

[Move test from 135.1-2023 into BTL Specified Tests and modify as shown]

7.3.2.47.1.7 Already Running Timer restarted with Default_Timeout

Reason for Change: Fix the property references.

Purpose: Verify the success of writes to Timer_Running with TRUE while already in the RUNNING state.

Test Concept: Configure and run the Timer T1 as necessary to put it into RUNNING state with an Initial_ValueTimeout different from Default_ValueTimeout. Then write the Timer_Running property with TRUE, and observe that Present_Value restarts with the value from Default_Timeout.

Configuration Requirements: T1 starts this test with the Timer_State equal to RUNNING. In service of observing the change between step 3 and step 6, it is necessary that at the test start, the Timer went into RUNNING state with an Initial_ValueTimeout different from Default_ValueTimeout.

Test Steps:

- 1. VERIFY Timer_State = RUNNING
- 2. READ DV = Default_Timeout
- 3. VERIFY Initial_Timeout <> DV
- 4. WRITE Timer_Running = TRUE
- 5. CHECK (IUT exhibits any changes configured in RUNNING_TO_RUNNING transition)

6. VERIFY Initial_Timeout = DV
7. VERIFY Present_Value ~= DV
8. VERIFY Timer_Running = TRUE
9. VERIFY Last_State_Change = RUNNING_TO_RUNNING

BTL-23.3 imp3-7: Resizing Group_Member_Names Test [BTLWG-1488]

Overview:

In this test, which is located in the document ANSI/ASHRAE Standard 135.1-2019, Step 11 is described as follows:

11. VERIFY Group_Members = (a BACnetDeviceObjectPropertyReference containing (Device, Instance number 4194303)), ARRAY INDEX = (some value from 3 through the value written in step 6)

This step does not consider the possible absents of the Device Instance.

The property "Group_Members" is of type BACnetARRAY of BACnetDeviceObjectPropertyReference. This type allows, in the case of an internal object, the Device Instance to be omitted and only the Object Identifier to be shown.

Furthermore, the standard states in Chapter 12.50.5.2:

"If the size of the Group_Members array is increased by writing to the size of either the Group_Members or Group_Member_Names property, the new array entries shall be initialized by setting the object or device instance numbers of the BACnetDeviceObjectPropertyReference equal to 4194303, indicating that the value is not initialized. The initial value of the other parameters is a local matter except that they shall be of the correct datatype."

Changes:

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, modify all occurrences of 135.1-2023 - 7.3.2.13.1 and 135.1-2023 - 7.3.2.13.2 to BTL - 7.3.2.13.1 and BTL - 7.3.2.13.2]

Specified Test Changes

[Move test 7.3.2.13.1 from 135.1 to BTL Specified Tests and modify as shown]

7.3.2.13.1 Resizing Group_Member_Names by Writing Group_Members Property Test

Reason for the change: Step 11 Object instance was not present in the test, reference: 153 2020- 12.50.5.2

Purpose: This test case verifies that when the size of the Group_Members array is changed by writing to it, the size of the Group_Member_Names and Present_Value arrays change accordingly and any new entries contain the specified initialized values. If the Group_Members array cannot be written, then this test shall not be performed.

Configuration Requirements: The IUT shall be configured with a Global Group object with a writable Group_Members property.

Test Concept: The Group_Members array is set to a certain size. It is then increased by writing the array size, decreased by writing the array, increased by writing the array and decreased by writing the array size. At each step the size of the Group_Member_Names and Present_Value arrays are verified and the initialized values of the new elements, if any, are checked.

Object1 shall be any Object-Type present in the IUT's Standard Object Types Supported, except object type Global-Group.

Test Steps:

1. TRANSMIT WriteProperty-Request,
 - 'Object Identifier' = (the Global Group object being tested),
 - 'Property Identifier' = Group_Members,
 - 'Property Array Index' = 0,
 - 'Property Value' = 2
2. RECEIVE Simple-ACK-PDU

3. VERIFY Group_Members = 2, ARRAY INDEX = 0
4. VERIFY Group_Member_Names = 2, ARRAY INDEX = 0
5. VERIFY Present_Value = 2, ARRAY INDEX = 0
6. TRANSMIT WriteProperty-Request,
 'Object Identifier' = (the Global Group object being tested),
 'Property Identifier' = Group_Members,
 'Property Array Index' = 0,
 'Property Value' = (some value greater than 2)
7. RECEIVE Simple-ACK-PDU
8. VERIFY Group_Members = (the value written in step 6), ARRAY INDEX = 0
9. VERIFY Group_Member_Names = (the value written in step 6), ARRAY INDEX = 0
10. VERIFY Present_Value = (the value written in step 6), ARRAY INDEX = 0
11. VERIFY Group_Members = (a BACnetDeviceObjectPropertyReference containing
 (Device, Instance number 4194303) | (Object1, 4194303)),
 ARRAY INDEX = (some value from 3 through the value written in step 6)
12. VERIFY Group_Member_Names = (an empty string),
 ARRAY INDEX = (some value from 3 through the value written in step 6)
13. VERIFY Present_Value = 'Access_Result' = PropertyAccessError (PROPERTY, VALUE_NOT_INITIALIZED),
 ARRAY INDEX = (some value from 3 through the value written in step 6)
14. TRANSMIT WriteProperty-Request,
 'Object Identifier' = (the Global Group object being tested),
 'Property Identifier' = Group_Members,
 'Property Value' = (a one-element array containing any valid BACnetDeviceObjectPropertyReference)
15. RECEIVE Simple-ACK-PDU
16. VERIFY Group_Members = 1, ARRAY INDEX = 0
17. VERIFY Group_Member_Names = 1, ARRAY INDEX = 0
18. VERIFY Present_Value = 1, ARRAY INDEX = 0
19. VERIFY Group_Members = (the array written in step 14)
20. TRANSMIT WriteProperty-Request,
 'Object Identifier' = (the Global Group object being tested),
 'Property Identifier' = Group_Members,
 'Property Value' = (an array of two or more valid BACnetDeviceObjectPropertyReference values)
21. RECEIVE Simple-ACK-PDU
22. VERIFY Group_Members = (the size of the array written in step 20), ARRAY INDEX = 0
23. VERIFY Group_Member_Names = (the size of the array written in step 20), ARRAY INDEX = 0
24. VERIFY Present_Value = (the size of the array written in step 20), ARRAY INDEX = 0
25. VERIFY Group_Members = (the array written in step 20)
26. TRANSMIT WriteProperty-Request,
 'Object Identifier' = (the Global Group object being tested),
 'Property Identifier' = Group_Members,
 'Property Array Index' = 0,
 'Property Value' = (some value between 0 and the size of the array written in step 20)
27. RECEIVE Simple-ACK-PDU
28. VERIFY Group_Members = (the size of the array written in step 26), ARRAY INDEX = 0
29. VERIFY Group_Member_Names = (the size of the array written in step 26), ARRAY INDEX = 0
30. VERIFY Present_Value = (the size of the array written in step 26), ARRAY INDEX = 0

[Move test 7.3.2.13.2 from 135.1 to BTL Specified Tests and modify as shown]

7.3.2.13.2 Resizing Group_Members by Writing Group_Member_Names Property Test

Reason for the change: Step 11 Object instance was not present in the test, reference: 153 2020- 12.50.5.2

Dependencies: WriteProperty Service Execution Tests, 9.22

Purpose: This test case verifies that when the size of the Group_Member_Names array is changed by writing to it, the size of the Group_Members and Present_Value arrays change accordingly and any new entries contain the specified initialized values. If the Group_Member_Names array cannot be written, then this test shall not be performed.

Configuration Requirements: The IUT shall be configured with a Global Group object with a writable Group_Member_Names property.

Test Concept: The Group_Member_Names array is set to a certain size. It is then increased by writing the array size, decreased by writing the array, increased by writing the array and decreased by writing the array size. At each step the size of the Group_Members and Present_Value arrays are verified and the initialized values of the new elements, if any, are checked. *Object1 shall be any Object-Type present in the IUT's Standard Object Types Supported, except object type Global-Group.*

Test Steps:

1. TRANSMIT WriteProperty-Request,
 'Object Identifier' = (the Global Group object being tested),
 'Property Identifier' = Group_Member_Names,
 'Property Array Index' = 0,
 'Property Value' = 2
2. RECEIVE Simple-ACK-PDU
3. VERIFY Group_Member_Names = 2, ARRAY INDEX = 0
4. VERIFY Group_Members = 2, ARRAY INDEX = 0
5. VERIFY Present_Value = 2, ARRAY INDEX = 0
6. TRANSMIT WriteProperty-Request,
 'Object Identifier' = (the Global Group object being tested),
 'Property Identifier' = Group_Member_Names,
 'Property Array Index' = 0,
 'Property Value' = (some value greater than 2)
7. RECEIVE Simple-ACK-PDU
8. VERIFY Group_Member_Names = (the value written in step 6), ARRAY INDEX = 0
9. VERIFY Group_Members = (the value written in step 6), ARRAY INDEX = 0
10. VERIFY Present_Value = (the value written in step 6), ARRAY INDEX = 0
11. VERIFY Group_Member_Names = (an empty string),
 ARRAY INDEX = (some value from 3 through the value written in step 6)
12. VERIFY Group_Members = (Device, Instance number 4194303) (Object1, 4194303),
 ARRAY INDEX = (some value from 3 through the value written in step 6)
13. VERIFY Present_Value = 'Access_Result' = PropertyAccessError (PROPERTY, VALUE_NOT_INITIALIZED),
 ARRAY INDEX = (some value from 3 through the value written in step 6)
14. TRANSMIT WriteProperty-Request,
 'Object Identifier' = (the Global Group object being tested),
 'Property Identifier' = Group_Member_Names,
 'Property Value' = (an array of one Character String)
15. RECEIVE Simple-ACK-PDU
16. VERIFY Group_Member_Names = 1, ARRAY INDEX = 0
17. VERIFY Group_Members = 1, ARRAY INDEX = 0
18. VERIFY Present_Value = 1, ARRAY INDEX = 0
19. VERIFY Group_Member_Names = (the array written in step 14)
20. TRANSMIT WriteProperty-Request,
 'Object Identifier' = (the Global Group object being tested),
 'Property Identifier' = Group_Member_Names,
 'Property Value' = (an array of two or more Character Strings)
21. RECEIVE Simple-ACK-PDU
22. VERIFY Group_Member_Names = (the size of the array written in step 20), ARRAY INDEX = 0
23. VERIFY Group_Members = (the size of the array written in step 20), ARRAY INDEX = 0
24. VERIFY Present_Value = (the size of the array written in step 20), ARRAY INDEX = 0
25. VERIFY Group_Member_Names = (the array of Character Strings written in step 20)
26. TRANSMIT WriteProperty-Request,
 'Object Identifier' = (the Global Group object being tested),
 'Property Identifier' = Group_Member_Names,
 'Property Array Index' = 0,
 'Property Value' = (some value between 0 and the size of the array written in step 20)
27. RECEIVE Simple-ACK-PDU
28. VERIFY Group_Member_Names = (the size of the array written in step 26), ARRAY INDEX = 0
29. VERIFY Group_Members = (the size of the array written in step 26), ARRAY INDEX = 0
30. VERIFY Present_Value = (the size of the array written in step 26), ARRAY INDEX = 0

BTL-23.3 imp3-8: WritePropertyMultiple Allowed Errors [BTLWG-1491]

Overview:

Interpretation Request clarified that a WritePropertyMultiple request that fails in its entirety can return either BACnet-Error-PDU or BACnet-Reject-PDU messages.

Changes:

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, Change reference to test 135.1-2023-9.23.2.16 to BTL in 4.8.1 WritePropertyMultiple-B Base Requirements section]

Specified Test Changes

9.23.2.16 WritePropertyMultiple Reject Test for first element of 'List of Write Access Specifications'

Reason for Change: Applied interpretation request for valid error codes.

Purpose: This test case verifies that if IUT does sends a Reject-PDU or Error-PDU then the write attempt for the remaining element of 'List of Write Access Specifications' do not take place.

Test Concept: Two writable properties, P1 having value X and P2 having value Y are written to the IUT but the portion of the WritePropertyMultiple specifying P1 is made invalid by omitting the 'Property Value' parameter. The value of the properties are checked to ensure that it has not changed.

Test Steps:

1. ~~VERIFY (O1), P1 = X~~ READ X = (O1), P1
2. ~~VERIFY (O2, P2 = Y)~~ READ Y = (O2), P2
3. TRANSMIT WritePropertyMultiple-Request,
 - 'Object Identifier' = O1,
 - 'Property Identifier' = P1,
 - 'Property Value' = (this field is missing including the opening and closing tags)
 - 'Object Identifier' = O2,
 - 'Property Identifier' = P2
 - 'Property Value' = (Any valid value not equal to Y))
4. RECEIVE WritePropertyMultiple-Error,
 - 'Error Class' = SERVICES,
 - 'Error Code' = INVALID_TAG,
 - 'Object Identifier' = O1
 - 'Property Identifier' = P1 |
 (RECEIVE BACnet-Reject-PDU,
 - 'Reject Reason' = INVALID_TAG |
 - MISSING_REQUIRED_PARAMETER |
 - INCONSISTENT_PARAMETERS |
 - INVALID_PARAMETER_DATA_TYPE |
 - ~~TOO_MANY_ARGUMENTS~~)
5. VERIFY (O1), P1 = X
6. VERIFY (O2), P2 = Y

BTL-23.3 imp3-9: Exclude Elevator Objects for DS-AM-A [BTLWG-1511]

Overview:

For 135-2020 / K.1.18 BIBB - Data Sharing-Advanced View-A (DS-AM-A), the 'Elevator objects' should be listed as exceptions in the directives.

Changes:

Checklist Changes

None

Test Plan Changes

4.14.1 Base Requirements

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

135.1-2023 - 8.22.4 - Accepting Input and Modifying Properties		
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test for all standard objects and properties, excluding the Life Safety, and Access Control, and Elevator objects, and the Object_Identifier and Object_Type properties. Also exclude any properties that are required to be read-only by the BACnet standard, and exclude properties which are commandable because those are covered by a different test. Repeat the test for a variety of values that cover the range of values required by the “Minimum Writable Value Ranges” table in the DS-M-A BIBB definition.
	Testing Hints	
...		

Specified Test Changes

None

BTL-23.3 imp3-10: Change Test Conditionality for Sections 4.2.1 and 4.4.1 in Test Plan [BTLWG-1516]**Overview:**

The Test Conditionality for tests in sections 4.2.1 and 4.4.1 aren't clear:

- Tests in 4.2.1 should have conditionality of “Must be executed using the ReadProperty service”
- Tests in 4.4.1 should have conditionality of “Must be executed using the ReadPropertyMultiple service”

Changes:**Checklist Changes**

None

Test Plan Changes

[Modify the Test Conditionality for all tests where ReadProperty service is not mentioned]

4.2 Data Sharing - ReadProperty - B**4.2.1 Base Requirements**

All devices must support this BIBB.

135.1-2023 – 7.1.1 – Read Support Test Procedure		
	Test Conditionality	Must be executed using the ReadProperty service. To satisfy this test item, this test needs only be executed using ReadProperty.
	Test Directives	
	Testing Hints	
135.1-2023 - 9.18.2.1 - Reading Non-Array Properties with an Array Index		
	Test Conditionality	Must be executed using the ReadProperty service.
	Test Directives	
	Testing Hints	
135.1-2023 - 9.18.2.3 - Reading an Unknown Object		
	Test Conditionality	Must be executed using the ReadProperty service.
	Test Directives	
	Testing Hints	
135.1-2023 - 9.18.2.4 - Reading an Unknown Property		
	Test Conditionality	Must be executed using the ReadProperty service.
	Test Directives	Be sure to test at least one property identifier that is within the ASHRAE allocated range for standard property identifiers, but that has not yet been defined.
	Testing Hints	
135.1-2023 - 9.18.1.3 - Reading a Property From the Device Object using the Unknown Instance		
	Test Conditionality	If the device implements protocol revision 4 or higher, this test must be executed using the ReadProperty service.
	Test Directives	
	Testing Hints	
135.1-2023 - 7.1.3 - Verifying Property List against the EPICS		
	Test Conditionality	Must be executed using the ReadProperty service if the IUT claims Protocol Revision 14 or greater.
	Test Directives	
	Testing Hints	
135.1-2023 - 9.18.1.7 - Reading Array Properties at different Array Indexes		
	Test Conditionality	Must be executed using the ReadProperty service.
	Test Directives	Repeat for all supported BACnetARRAY properties
	Testing Hints	

[Modify the Test Conditionality for all tests where ReadPropertyMultiple service is not mentioned]

4.4 Data Sharing - ReadPropertyMultiple - B

4.4.1 Base Requirements

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

135.1-2023 – 7.1.1 – Read Support Test Procedure		
	Test Conditionality	Must be executed using the ReadPropertyMultiple service. To satisfy this test item, test 7.1 need only be executed using ReadPropertyMultiple.
	Test Directives	
	Testing Hints	
135.1-2023 - 9.20.1.1 - Reading a Single Property from a Single Object		
	Test Conditionality	Must be executed using the ReadPropertyMultiple service.
	Test Directives	
	Testing Hints	
135.1-2023 - 9.20.1.2 - Reading Multiple Properties from a Single Object		
	Test Conditionality	Must be executed using the ReadPropertyMultiple service.
	Test Directives	
	Testing Hints	
135.1-2023 - 9.20.1.3 - Reading a Single Property from Multiple Objects		
	Test Conditionality	Must be executed using the ReadPropertyMultiple service. This test can be skipped if the IUT cannot be made to contain more than 1 object.
	Test Directives	
	Testing Hints	
135.1-2023 - 9.20.1.4 - Reading Multiple Properties from Multiple Objects		
	Test Conditionality	Must be executed using the ReadPropertyMultiple service. This test can be skipped if the IUT cannot be made to contain more than 1 object.
	Test Directives	
	Testing Hints	
135.1-2023 - 9.20.1.5 - Reading Multiple Properties with a Single Embedded Access Error		
	Test Conditionality	Must be executed using the ReadPropertyMultiple service.
	Test Directives	
	Testing Hints	
135.1-2023 - 9.20.1.6 - Reading Multiple Properties with Multiple Embedded Access Errors		
	Test Conditionality	Must be executed using the ReadPropertyMultiple service.
	Test Directives	
	Testing Hints	
135.1-2023 - 9.20.1.7 - Reading ALL Properties		
	Test Conditionality	Must be executed using the ReadPropertyMultiple service. This test shall be skipped for any object type whose set of properties cannot be transmitted in the largest supported response message based on the IUT's APDU and segmentation limitations.
	Test Directives	
	Testing Hints	
135.1-2023 - 9.20.1.8 - Reading OPTIONAL Properties		
	Test Conditionality	Must be executed using the ReadPropertyMultiple service. This test shall be skipped for any object type whose set of optional properties cannot be transmitted in the largest supported response message based on the IUT's APDU and segmentation limitations.
	Test Directives	
	Testing Hints	The pre-tester should apply this test to every object type.
135.1-2023 - 9.20.1.9 - Reading REQUIRED Properties		
	Test Conditionality	Must be executed using the ReadPropertyMultiple service. This test shall be skipped for any object type whose set of optional properties cannot be

		transmitted in the largest supported response message based on the IUT's APDU and segmentation limitations.
	Test Directives	
	Testing Hints	The pre-tester should apply this test to every object type. If the set of properties differs between instances of the same object type in the IUT, each form of the object type should be tested.
135.1-2023 - 9.20.1.10 - Reading the Size of an Array		
	Test Conditionality	Must be executed using the ReadPropertyMultiple service.
	Test Directives	
	Testing Hints	
135.1-2023 - 9.20.2.1 - Reading a Single, Unsupported Property from a Single Object		
	Test Conditionality	Must be executed using the ReadPropertyMultiple service.
	Test Directives	
	Testing Hints	
135.1-2023 - 9.20.2.2 - Reading Multiple Properties with Access Errors for Every Property		
	Test Conditionality	Must be executed using the ReadPropertyMultiple service.
	Test Directives	
	Testing Hints	
135.1-2023 - 9.20.2.3 - Reading Non-Array Properties with an Array Index		
	Test Conditionality	Must be executed using the ReadPropertyMultiple service.
	Test Directives	
	Testing Hints	
135.1-2023 - 9.20.1.11 - Reading a Property From the Device Object using the Unknown Instance		
	Test Conditionality	If the device implements protocol revision 4 or higher, this test must be executed using the ReadPropertyMultiple service. If the device does not support ReadPropertyMultiple, this test may be skipped.
	Test Directives	
	Testing Hints	
BTL - 9.20.1.X2 - ReadPropertyMultiple Array Properties		
	Test Conditionality	Must be executed using the ReadPropertyMultiple service.
	Test Directives	Repeat for all supported BACnetARRAY properties
	Testing Hints	

Specified Test Changes

None

BTL-23.3 imp3-11: Clarify DS-WP-B and DS-WPM-B Test Directives [BTLWG-1526]

Overview:

Issue #1

Test 7.2.2 states the test is performed using WP and WPM but referenced separately in DS-WP-B and DS-WPM-B. This causes confusion particularly if the IUT only supports WP.

Issue #2

Test 7.2.2 is a general test that checks that all writable properties in the IUT's database are actually writeable and tests multiple values. This test contains Notes to Tester to deal with properties that contain internal processes and conditionally writable properties.

Tests 9.22.1.5 and 9.23.1.8 test the writability of properties using WP or WPM respectively, does not test multiple values, and contains no conditionality.

The Test Plan allows 7.2.2 to be skipped if 9.22.1.5 or 9.23.1.8 is run. The Test Directives for 7.2.2, 9.22.1.5, and 9.23.1.8 are confusing.

Changes:

Checklist Changes

BTL Checklist Changes

Data Sharing - WriteProperty - B		
	R	Base Requirements
	C ¹	Contains writable non-array properties
	C ¹	Contains writable array properties
	O	Contains resizable array properties
	C ^{2,3}	Contains writable list properties
	O	Contains commandable properties
	OC ²	Contains non-commandable properties which accept a written NULL value
	C ²	Contains writable BOOLEAN properties
	C ²	Contains writable Enumerated properties
	C ²	Contains writable INTEGER properties
	C ²	Contains writable Unsigned properties
	C ²	Contains writable REAL properties
	C ²	Contains writable Double properties
	C ²	Contains writable Time properties
	C ²	Contains writable Date properties
	C ²	Contains writable Character String properties
	C ²	Contains writable Octet String properties
	C ²	Contains writable Bit String properties
	C ²	Contains writable BACnetObjectIdentifier properties
	C ²	Contains writable properties with non-basic data types
	C ²	Contains writable proprietary properties with basic data types
¹ At least one of these options is required in order to claim conformance to this BIBB.		
² At least one of these options is required in order to claim conformance to this BIBB.		
³ Required if the device contains properties modifiable via AddListElement/RemoveListElement.		

Data Sharing - WritePropertyMultiple - B		
	R	Base Requirements
	C ¹	Contains multiple objects with writable properties
	C ¹	Contains objects with multiple writable properties

	O	Contains multiple objects with multiple writable properties
	O	Contains writable non-array properties
	O	Contains writable array properties
	O	Contains resizable array properties
	C ³	Contains writable list properties
	O	Contains commandable properties
	O	Contains non-commandable properties which accept a written NULL value
	OC ²	Contains writable BOOLEAN properties
	OC ²	Contains writable Enumerated properties
	OC ²	Contains writable INTEGER properties
	OC ²	Contains writable Unsigned properties
	OC ²	Contains writable REAL properties
	OC ²	Contains writable Double properties
	OC ²	Contains writable Time properties
	OC ²	Contains writable Date properties
	OC ²	Contains writable Character String properties
	OC ²	Contains writable Octet String properties
	OC ²	Contains writable Bit String properties
	OC ²	Contains writable BACnetObjectIdentifier properties
	OC ²	Contains writable properties with non-basic data types
	OC ²	Contains writable proprietary properties with basic data types
¹ At least one of these options is required in order to claim conformance to this BIBB.		
² At least one of these options is required in order to claim conformance to this BIBB.		
²³ Required if the device contains properties modifiable via AddListElement/RemoveListElement.		

Test Plan Changes

4.6 Data Sharing - WriteProperty - B

4.6.1 Base Requirements

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

BTL - 7.2.2 - Write Support Test Procedure		
	Test Conditionality	Must be executed.
	Test Directives	Run this test using the WriteProperty Service.
	Testing Hints	
...		

4.6.7 Contains non-commandable Properties which Accept a Written NULL Value

The IUT contains, or can be made to contain, a writable property that accepts a written NULL value.

135.1-2023 - 9.22.1.5 - Writing to Properties Based on Data Type		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	Schedule_Default and Present_Value of the Schedule Object, Alarm_Values and Fault_Values of the CharacterString Value Object and Low_Diff_Limit in the Loop Object are standard properties that should accept a written NULL.

4.6.8 Contains Writable BOOLEAN Properties

The IUT contains, or can be made to contain, a writable property with a data type of BOOLEAN.

Verify Test Selection	135.1-2023 - 9.22.1.5 - Writing to Properties Based on Data Type
-----------------------	--

	Test Conditionality	Must be executed.
	Test Directives	Ensure test BTL - 7.2.2 is executed on at least one property of the specified data type using the WriteProperty Service.
	Testing Hints	

4.6.9 Contains Writable Enumerated Properties

The IUT contains, or can be made to contain, a writable property with a data type of Enumerated.

Verify Test Selection 135.1-2023 – 9.22.1.5 – Writing to Properties Based on Data Type		
	Test Conditionality	Must be executed.
	Test Directives	Ensure test BTL - 7.2.2 is executed on at least one property of the specified data type using the WriteProperty Service.
	Testing Hints	

4.6.10 Contains Writable INTEGER Properties

The IUT contains, or can be made to contain, a writable property with a data type of INTEGER.

Verify Test Selection 135.1-2023 – 9.22.1.5 – Writing to Properties Based on Data Type		
	Test Conditionality	Must be executed.
	Test Directives	Ensure test BTL - 7.2.2 is executed on at least one property of the specified data type using the WriteProperty Service.
	Testing Hints	

4.6.11 Contains Writable Unsigned Properties

The IUT contains, or can be made to contain, a writable property with a data type of Unsigned.

Verify Test Selection 135.1-2023 – 9.22.1.5 – Writing to Properties Based on Data Type		
	Test Conditionality	Must be executed.
	Test Directives	Ensure test BTL - 7.2.2 is executed on at least one property of the specified data type using the WriteProperty Service.
	Testing Hints	

4.6.12 Contains Writable REAL Properties

The IUT contains, or can be made to contain, a writable property with a data type of REAL.

Verify Test Selection 135.1-2023 – 9.22.1.5 – Writing to Properties Based on Data Type		
	Test Conditionality	Must be executed.
	Test Directives	Ensure test BTL - 7.2.2 is executed on at least one property of the specified data type using the WriteProperty Service.
	Testing Hints	

4.6.13 Contains Writable Double Properties

The IUT contains, or can be made to contain, a writable property with a data type of Double.

Verify Test Selection 135.1-2023 – 9.22.1.5 – Writing to Properties Based on Data Type		
	Test Conditionality	Must be executed.
	Test Directives	Ensure test BTL - 7.2.2 is executed on at least one property of the specified data type using the WriteProperty Service.
	Testing Hints	

4.6.14 Contains Writable Time Properties

The IUT contains, or can be made to contain, a writable property with a data type of Time.

Verify Test Selection 135.1-2023 – 9.22.1.5 – Writing to Properties Based on Data Type		
	Test Conditionality	Must be executed.
	Test Directives	Ensure test BTL - 7.2.2 is executed on at least one property of the specified data type using the WriteProperty Service.
	Testing Hints	

	Testing Hints	
--	---------------	--

4.6.15 Contains Writable Date Properties

The IUT contains, or can be made to contain, a writable property with a data type of Date.

Verify Test Selection 135.1-2023 - 9.22.1.5 - Writing to Properties Based on Data Type		
	Test Conditionality	Must be executed.
	Test Directives	Ensure test BTL - 7.2.2 is executed on at least one property of the specified data type using the WriteProperty Service.
	Testing Hints	

4.6.16 Contains Writable Character String Properties

The IUT contains, or can be made to contain, a writable property with a data type of Character String.

Verify Test Selection 135.1-2023 - 9.22.1.5 - Writing to Properties Based on Data Type		
	Test Conditionality	Must be executed.
	Test Directives	Ensure test BTL - 7.2.2 is executed on at least one property of the specified data type using the WriteProperty Service.
	Testing Hints	

4.6.17 Contains Writable Octet String Properties

The IUT contains, or can be made to contain, a writable property with a data type of Octet String.

Verify Test Selection 135.1-2023 - 9.22.1.5 - Writing to Properties Based on Data Type		
	Test Conditionality	Must be executed.
	Test Directives	Ensure test BTL - 7.2.2 is executed on at least one property of the specified data type using the WriteProperty Service.
	Testing Hints	

4.6.18 Contains Writable Bit String Properties

The IUT contains, or can be made to contain, a writable property with a data type of Bit String.

Verify Test Selection 135.1-2023 - 9.22.1.5 - Writing to Properties Based on Data Type		
	Test Conditionality	Must be executed.
	Test Directives	Ensure test BTL - 7.2.2 is executed on at least one property of the specified data type using the WriteProperty Service.
	Testing Hints	

4.6.19 Contains Writable BACnetObjectIdentifier Properties

The IUT contains, or can be made to contain, a writable property with a data type of BACnetObjectIdentifier.

Verify Test Selection 135.1-2023 - 9.22.1.5 - Writing to Properties Based on Data Type		
	Test Conditionality	Must be executed.
	Test Directives	Ensure test BTL - 7.2.2 is executed on at least one property of the specified data type using the WriteProperty Service.
	Testing Hints	

4.6.20 Contains Writable Properties with Non-Basic Data Types

The IUT contains, or can be made to contain, a writable property with a non-basic data type. A non-basic data type is one that is represented by a SEQUENCE or CHOICE construct when described in ASN.1.

Verify Test Selection 135.1-2023 - 9.22.1.5 - Writing to Properties Based on Data Type		
	Test Conditionality	Must be executed.
	Test Directives	Ensure test BTL - 7.2.2 is executed on at least one property of a non-basic data type using the WriteProperty Service.
	Testing Hints	

4.8 Data Sharing - WritePropertyMultiple - B

4.8.1 Base Requirements

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

BTL - 7.2.2 - Write Support Test Procedure		
	Test Conditionality	Must be executed.
	Test Directives	Run this test using the WritePropertyMultiple Service.
	Testing Hints	
...		

4.8.5 Contains Writable Non-Array Properties

The IUT contains, or can be made to contain, a writable non-array property.

Verify Test Selection 135.1-2023 - 9.23.1.8 - Writing to Properties Based on Data Type		
	Test Conditionality	Must be executed.
	Test Directives	Ensure test BTL - 7.2.2 is executed on a non-array property of any data type..
	Testing Hints	

4.8.10 Contains non-commandable Properties which Accept a Written NULL Value

The IUT contains, or can be made to contain, a writable property that accepts a written NULL value.

Verify Test Selection 135.1-2023 - 9.23.1.8 - Writing to Properties Based on Data Type		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	Schedule_Default and Present_Value of the Schedule Object, Alarm_Values and Fault_Values of the CharacterString Value Object and Low_Diff_Limit in the Loop Object are standard properties that should accept a written NULL.

4.8.11 Contains Writable BOOLEAN Properties

The IUT contains, or can be made to contain, a writable property with a data type of BOOLEAN.

Verify Test Selection 135.1-2023 - 9.23.1.8 - Writing to Properties Based on Data Type		
	Test Conditionality	Must be executed.
	Test Directives	Ensure test BTL - 7.2.2 is executed on at least one property of the specified data type using the WritePropertyMultiple Service.
	Testing Hints	

4.8.12 Contains Writable Enumerated Properties

The IUT contains, or can be made to contain, a writable property with a data type of Enumerated.

Verify Test Selection 135.1-2023 - 9.23.1.8 - Writing to Properties Based on Data Type		
	Test Conditionality	Must be executed.
	Test Directives	Ensure test BTL - 7.2.2 is executed on at least one property of the specified data type using the WritePropertyMultiple Service.
	Testing Hints	

4.8.13 Contains Writable INTEGER Properties

The IUT contains, or can be made to contain, a writable property with a data type of INTEGER.

Verify Test Selection 135.1-2023 - 9.23.1.8 - Writing to Properties Based on Data Type		
	Test Conditionality	Must be executed.
	Test Directives	Ensure test BTL - 7.2.2 is executed on at least one property of the specified data type using the WritePropertyMultiple Service.

	Testing Hints	
--	---------------	--

4.8.14 Contains Writable Unsigned Properties

The IUT contains, or can be made to contain, a writable property with a data type of Unsigned.

Verify Test Selection 135.1-2023-9.23.1.8 – Writing to Properties Based on Data Type		
	Test Conditionality	Must be executed.
	Test Directives	Ensure test BTL - 7.2.2 is executed on at least one property of the specified data type using the WritePropertyMultiple Service.
	Testing Hints	

4.8.15 Contains Writable REAL Properties

The IUT contains, or can be made to contain, a writable property with a data type of REAL.

Verify Test Selection 135.1-2023-9.23.1.8 – Writing to Properties Based on Data Type		
	Test Conditionality	Must be executed.
	Test Directives	Ensure test BTL - 7.2.2 is executed on at least one property of the specified data type using the WritePropertyMultiple Service.
	Testing Hints	

4.8.16 Contains Writable Double Properties

The IUT contains, or can be made to contain, a writable property with a data type of Double.

Verify Test Selection 135.1-2023-9.23.1.8 – Writing to Properties Based on Data Type		
	Test Conditionality	Must be executed.
	Test Directives	Ensure test BTL - 7.2.2 is executed on at least one property of the specified data type using the WritePropertyMultiple Service.
	Testing Hints	

4.8.17 Contains Writable Time Properties

The IUT contains, or can be made to contain, a writable property with a data type of Time.

Verify Test Selection 135.1-2023-9.23.1.8 – Writing to Properties Based on Data Type		
	Test Conditionality	Must be executed.
	Test Directives	Ensure test BTL - 7.2.2 is executed on at least one property of the specified data type using the WritePropertyMultiple Service.
	Testing Hints	

4.8.18 Contains Writable Date Properties

The IUT contains, or can be made to contain, a writable property with a data type of Date.

Verify Test Selection 135.1-2023-9.23.1.8 – Writing to Properties Based on Data Type		
	Test Conditionality	Must be executed.
	Test Directives	Ensure test BTL - 7.2.2 is executed on at least one property of the specified data type using the WritePropertyMultiple Service.
	Testing Hints	

4.8.19 Contains Writable Character String Properties

The IUT contains, or can be made to contain, a writable property with a data type of Character String.

Verify Test Selection 135.1-2023-9.23.1.8 – Writing to Properties Based on Data Type		
	Test Conditionality	Must be executed.
	Test Directives	Ensure test BTL - 7.2.2 is executed on at least one property of the specified data type using the WritePropertyMultiple Service.
	Testing Hints	

4.8.20 Contains Writable Octet String Properties

The IUT contains, or can be made to contain, a writable property with a data type of Octet String.

Verify Test Selection 135.1 - 2023 - 9.23.1.8 Writing to Properties Based on Data Type		
	Test Conditionality	Must be executed.
	Test Directives	Ensure test BTL - 7.2.2 is executed on at least one property of the specified data type using the WritePropertyMultiple Service.
	Testing Hints	

4.8.21 Contains Writable Bit String Properties

The IUT contains, or can be made to contain, a writable property with a data type of Bit String.

Verify Test Selection 135.1 - 2023 - 9.23.1.8 Writing to Properties Based on Data Type		
	Test Conditionality	Must be executed.
	Test Directives	Ensure test BTL - 7.2.2 is executed on at least one property of the specified data type using the WritePropertyMultiple Service.
	Testing Hints	

4.8.22 Contains Writable BACnetObjectIdentifier Properties

The IUT contains, or can be made to contain, a writable property with a data type of BACnetObjectIdentifier.

Verify Test Selection 135.1 - 2023 - 9.23.1.8 Writing to Properties Based on Data Type		
	Test Conditionality	Must be executed.
	Test Directives	Ensure test BTL - 7.2.2 is executed on at least one property of the specified data type using the WritePropertyMultiple Service.
	Testing Hints	

4.8.23 Contains Writable Properties with Non-Basic Data Types

The IUT contains, or can be made to contain, a writable property with a non-basic data type. A non-basic data type is one that is represented by a SEQUENCE or CHOICE construct when described in ASN.1.

Verify Test Selection 135.1 - 2023 - 9.23.1.8 Writing to Properties Based on Data Type		
	Test Conditionality	Must be executed.
	Test Directives	Ensure test BTL - 7.2.2 is executed on at least one property of a non-basic data type using the WritePropertyMultiple Service.
	Testing Hints	

Specified Test Changes

[Change test 135.1 - 2023 - 7.2.2]

7.2.2 Write Support Test Procedure

Reason For Change: Remove specific WriteProperty and WritePropertyMultiple service requirements.

Purpose: To verify that all writable properties of all objects can be written ~~to using BACnet WriteProperty and WritePropertyMultiple services. The test is performed once using WriteProperty and once using WritePropertyMultiple. When writing to array properties, the whole array shall be written without using an array index, where possible.~~

Test Concept: Each writable property is written multiple times verifying the writable range. After each write, the value is verified to have been updated in the property. ~~The test is performed once using WriteProperty and once using WritePropertyMultiple.~~ When writing to array properties, the whole array shall be written without using an array index, where possible.

Notes to Tester: An internal process may set the ~~Present_Value~~ *value* of some properties ~~to a value different from the written back to the default~~ value after a successful write, as in the case of a momentary pushbutton, or the Record_Count property. For properties that exhibit this type of behavior, skip the VERIFY step.

Notes to Tester: When a property is currently not writable, the IUT shall return an Error-PDU with 'Error Class' = PROPERTY and 'Error Code' = WRITE_ACCESS_DENIED.

Notes to Tester: Do not run this test against any properties in the Network Port objects and against the Object_Identifier of the Device object.

Test Steps:

1. REPEAT X = (all objects in the IUT's database, except *as specified in the Notes to Tester* ~~Network Port objects~~) DO {
 - REPEAT Y = (all writable properties in object X) DO {
 - REPEAT Z = (all values meeting the functional range requirements of 7.2.1, and any additional restrictions placed on the allowable property values by the vendor) DO {
 - WRITE (X), Y = Z,
 - VERIFY (X), Y = Z

BTL-23.3 imp3-12: AE-N-E-B Removal of Duplicate Entry [BTLWG-512]

Overview:

Remove 'Supports Event_Message_Texts property' checklist item as already in AE-N-I-B.

Changes:

Checklist Changes

[In BTL Checklist, remove the 'Supports Event_Message_Texts property' entry from AE-N-E-B. No other changes to this section.]

Alarm and Event Management - Notification - External - B		

	Q	Supports Event_Message_Texts property
<div><div>¹ At least one of these options must be supported to claim support for this BIBB. It is recommended that a standard BACnet algorithm be used instead of a proprietary algorithm whenever possible.</div><div>² Contact BTL for interim tests for this algorithm.</div><div>³ Protocol_Revision 16 or higher must be claimed.</div><div>⁴ Protocol_Revision 17 or higher must be claimed.</div><div>⁵ Protocol_Revision 18 or higher must be claimed.</div><div>⁶ A device shall support CHANGE_OF_RELIABILITY in any object which generates event notifications and in which the Reliability property can take on a value other than NO_FAULT_DETECTED.</div></div>		

Test Plan Changes

[In BTL Test Plan, remove section 5.3.32 (same section as removed from checklist above).]

~~5.3.32 Supports Event_Message_Texts Property~~

~~The IUT contains one or more objects that support the Event_Message_Texts property.~~

135.1 2023 7.3.1.17 Event_Message_Texts Tests		
	Test Conditionality	Must be executed.
	Test Directives	Repeat test once for each object type in the IUT that contains an Event_Message_Texts property.
	Testing Hints	

Specified Test Changes

None

BTL-23.3 imp3-13: Make Intrinsic Reporting Consistent [BTLWG-1312]

Overview:

The checklist for Intrinsic Reporting is not consistent and does not explicitly allow for CHANGE_OF_RELIABILITY reporting.

Changes:

Checklist Changes

[Add to each Clause 3 object specified]

	O	Supports Intrinsic Reporting
--	---	------------------------------

- Analog Input
- Analog Output
- Analog Value
- Binary Input
- Binary Output
- Binary Value
- Command
- Device
- Loop
- Multi-State Input
- Multi-State Output
- Multi-State Value
- Notification Class
- Schedule
- Bitstring Value
- Characterstring Value
- Date Pattern Value
- Date Value
- Datetime Pattern Value
- Datetime Value
- Integer Value
- Large Analog Value
- Positive Integer Value
- Time Pattern Value
- Time Value
- Global Group
- Accumulator
- Program
- Life Safety Point
- Life Safety Zone
- Pulse Converter
- Load Control
- Access Point
- Access Zone
- Credential Data Input
- Channel
- Binary Lighting Output
- Network Port
- Timer
- Lift
- Escalator
- Staging
- Audit Reporter
- Audit Log

Note, Access Door and Lighting Output objects already have this Checklist item:

Test Plan Changes

[Add]

3.1.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Analog Input objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with at least one of "Implements the OUT_OF_RANGE algorithm", "Implements the CHANGE_OF_RELIABILITY - FAULT_OUT_OF_RANGE algorithm", or "Implements the CHANGE_OF_RELIABILITY – NONE".
	Testing Hints	

3.2.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Analog Output objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with at least one of "Implements the OUT_OF_RANGE algorithm", or "Implements the CHANGE_OF_RELIABILITY – NONE".
	Testing Hints	

3.3.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Analog Value objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with at least one of "Implements the OUT_OF_RANGE algorithm", "Implements the CHANGE_OF_RELIABILITY - FAULT_OUT_OF_RANGE algorithm", or "Implements the CHANGE_OF_RELIABILITY – NONE".
	Testing Hints	

3.5.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Binary Input objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with at least one of "Implements the CHANGE_OF_STATE algorithm", or "Implements the CHANGE_OF_RELIABILITY – NONE".
	Testing Hints	

3.6.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Binary Output objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with at least one of "Implements the COMMAND_FAILURE algorithm", or "Implements the CHANGE OF RELIABILITY – NONE".
	Testing Hints	

3.7.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Binary Value objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with at least one of "Implements the CHANGE_OF_STATE algorithm", or "Implements the CHANGE OF RELIABILITY – NONE".
	Testing Hints	

3.9.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Command objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with option "Implements the CHANGE OF RELIABILITY – NONE".
	Testing Hints	

3.10.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in the Device object.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with option "Implements the CHANGE_OF_RELIABILITY – NONE".
	Testing Hints	

3.13.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Loop objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with at least one of "Implements the FLOATING_LIMIT algorithm", or "Implements the CHANGE OF RELIABILITY – NONE".
	Testing Hints	

3.14.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Multi-State Input objects.

Verify Checklist		
	Test Conditionality	Must be executed.

	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with at least one of "Implements the CHANGE_OF_STATE algorithm", "Implements the CHANGE_OF_RELIABILITY - FAULT_STATE algorithm", or "Implements the CHANGE_OF_RELIABILITY – NONE".
	Testing Hints	

3.15.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Multi-State Output objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with at least one of "Implements the COMMAND_FAILURE algorithm", or "Implements the CHANGE_OF_RELIABILITY – NONE".
	Testing Hints	

3.16.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Multi-State Value objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with at least one of "Implements the CHANGE_OF_STATE algorithm", "Implements the CHANGE_OF_RELIABILITY - FAULT_STATE algorithm", or "Implements the CHANGE_OF_RELIABILITY – NONE".
	Testing Hints	

3.17.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Notification Class objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with option "Implements the CHANGE_OF_RELIABILITY – NONE".
	Testing Hints	

3.19.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Schedule objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with option "Implements the CHANGE_OF_RELIABILITY – NONE".
	Testing Hints	

3.24.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Bitstring Value objects.

Verify Checklist		
	Test Conditionality	Must be executed.

	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with at least one of "Implements the CHANGE_OF_BITSTRING algorithm", or "Implements the CHANGE_OF_RELIABILITY – NONE".
	Testing Hints	

3.25.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in CharacterString Value objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with at least one of "Implements the CHANGE_OF_CHARACTERSTRING algorithm", or "Implements the CHANGE_OF_RELIABILITY – NONE".
	Testing Hints	

3.26.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Date Pattern Value objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with option "Implements the CHANGE OF RELIABILITY – NONE".
	Testing Hints	

3.27.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Date Value objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with option "Implements the CHANGE OF RELIABILITY – NONE".
	Testing Hints	

3.28.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Datetime Pattern Value objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with option "Implements the CHANGE OF RELIABILITY – NONE".
	Testing Hints	

3.29.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Datetime Value objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with option "Implements the CHANGE OF RELIABILITY – NONE".
	Testing Hints	

3.30.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Integer Value objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with at least one of "Implements the SIGNED_OUT_OF_RANGE algorithm", "Implements the CHANGE_OF_RELIABILITY - FAULT_OUT_OF_RANGE algorithm", or "Implements the CHANGE_OF_RELIABILITY - NONE".
	Testing Hints	

3.31.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Large Analog Value objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with at least one of "Implements the DOUBLE_OUT_OF_RANGE algorithm", "Implements the CHANGE_OF_RELIABILITY - FAULT_OUT_OF_RANGE algorithm", or "Implements the CHANGE_OF_RELIABILITY - NONE".
	Testing Hints	

3.33.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Positive Integer Value objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with at least one of "Implements the UNSIGNED_OUT_OF_RANGE algorithm", "Implements the CHANGE_OF_RELIABILITY - FAULT_OUT_OF_RANGE algorithm", or "Implements the CHANGE_OF_RELIABILITY - NONE".
	Testing Hints	

3.34.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Time Pattern Value objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with option "Implements the CHANGE_OF_RELIABILITY - NONE".
	Testing Hints	

3.35.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Time Value objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with option "Implements the CHANGE_OF_RELIABILITY - NONE".
	Testing Hints	

3.36.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Global Group objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with at least one of "Implements the CHANGE_OF_STATUS_FLAGS algorithm", "Implements the CHANGE_OF_RELIABILITY - FAULT_STATUS_FLAGS algorithm", or "Implements the CHANGE_OF_RELIABILITY - NONE".
	Testing Hints	

3.37.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Accumulator objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with at least one of "Implements the UNSIGNED_RANGE algorithm", "Implements the CHANGE_OF_RELIABILITY - FAULT_OUT_OF_RANGE algorithm", or "Implements the CHANGE_OF_RELIABILITY - NONE".
	Testing Hints	

3.38.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Program objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with option "Implements the CHANGE_OF_RELIABILITY - NONE".
	Testing Hints	

3.39.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Life Safety Point objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with at least one of "Implements the CHANGE_OF_LIFE_SAFETY algorithm", "Implements the CHANGE_OF_RELIABILITY - FAULT_LIFE_SAFETY algorithm", or "Implements the CHANGE_OF_RELIABILITY - NONE".
	Testing Hints	

3.40.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Life Safety Zone objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with at least one of "Implements the CHANGE_OF_LIFE_SAFETY algorithm", "Implements the CHANGE_OF_RELIABILITY -

		FAULT_LIFE_SAFETY algorithm”, or “Implements the CHANGE_OF_RELIABILITY – NONE”.
	Testing Hints	

3.41.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Pulse Converter objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with at least one of "Implements the OUT_OF_RANGE algorithm”, or “Implements the CHANGE_OF_RELIABILITY – NONE”.
	Testing Hints	

3.43.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Load Control objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with at least one of "Implements the CHANGE_OF_STATE algorithm”, or “Implements the CHANGE_OF_RELIABILITY – NONE”.
	Testing Hints	

3.44.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Access Point objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with at least one of "Implements the ACCESS_EVENT algorithm”, or “Implements the CHANGE_OF_RELIABILITY – NONE”.
	Testing Hints	

3.45.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Access Zone objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with at least one of "Implements the CHANGE_OF_STATE algorithm”, or “Implements the CHANGE_OF_RELIABILITY – NONE”.
	Testing Hints	

3.49.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Credential Data Input objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with option "Implements the CHANGE_OF_RELIABILITY – NONE”.
	Testing Hints	

3.53.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Channel objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with option "Implements the CHANGE_OF_RELIABILITY – NONE".
	Testing Hints	

3.55.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Binary Lighting Output objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with option "Implements the CHANGE_OF_RELIABILITY – NONE".
	Testing Hints	

3.56.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Network Port objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with option "Implements the CHANGE_OF_RELIABILITY – NONE".
	Testing Hints	

3.57.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Timer objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with at least one of "Implements the CHANGE_OF_TIMER algorithm", or "Implements the CHANGE_OF_RELIABILITY – NONE".
	Testing Hints	

3.59.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Lift objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with at least one of "Implements the CHANGE_OF_STATE algorithm", "Implements the CHANGE_OF_RELIABILITY - FAULT_LISTED algorithm", or "Implements the CHANGE_OF_RELIABILITY – NONE".
	Testing Hints	

3.60.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Escalator objects.

Verify Checklist		
------------------	--	--

	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with at least one of "Implements the CHANGE_OF_STATE algorithm", "Implements the CHANGE_OF_RELIABILITY - FAULT_LISTED algorithm", or "Implements the CHANGE_OF_RELIABILITY – NONE".
	Testing Hints	

3.62.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Staging objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with option "Implements the CHANGE OF RELIABILITY – NONE".
	Testing Hints	

3.63.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Audit Reporter objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with option "Implements the CHANGE OF RELIABILITY – NONE".
	Testing Hints	

3.64.X Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Audit Log objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with option "Implements the CHANGE OF RELIABILITY – NONE".
	Testing Hints	

[Change]

3.42.9 Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Access Door objects.

135.1-2023 - 7.3.2.40.1.8 - Door Open Too Long Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with at least one of "Implements the CHANGE OF STATE algorithm", "Implements the CHANGE OF RELIABILITY - FAULT STATE algorithm", or "Implements the CHANGE_OF_RELIABILITY – NONE".
	Testing Hints	

3.54.13 Supports Intrinsic Reporting

The IUT supports intrinsic reporting in Lighting Output objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-I-B in the Checklist with option "Implements the CHANGE_OF_RELIABILITY - NONE".
	Testing Hints	

Specified Test Changes

None

BTL-23.3 imp3-14: Add COV A and B Support for Other Standard Object Types [BTLWG-1373]**Overview:**

Clause 13.1 states: "The different standard objects that support standardized COV reporting use different criteria for determining that a "change of value" has occurred, which are summarized in Table 13-1. Proprietary object types, or other standard object types not listed in Table 13-1, that support COV reporting of the Present_Value property, should follow these criteria whenever possible. Any objects that may optionally provide COV or COV-multiple support and the change of value algorithms they shall employ are summarized in Tables 13-1 and 13-1a."

Note, due to limited interoperability benefit this proposal does not include proprietary object types.

Changes:**Checklist Changes**

[In BTL Checklist, add the two sections highlighted below.]

Data Sharing - Change Of Value - A		
	R	Base Requirements
	R	Subscribes with lifetimes up to 8 hours in duration
	C ¹	Can subscribe for confirmed notifications
	C ¹	Can subscribe for unconfirmed notifications
	C ²	Can subscribe for COV from Analog objects
	C ²	Can subscribe for COV from Binary objects
	C ²	Can subscribe for COV from Life Safety objects
	C ²	Can subscribe for COV from Loop objects
	C ²	Can subscribe for COV from Multi-state objects
	C ²	Can subscribe for COV from CharacterString objects
	C ²	Can subscribe for COV from Date objects
	C ²	Can subscribe for COV from DateTime objects
	C ²	Can subscribe for COV from Integer objects
	C ²	Can subscribe for COV from Large Analog objects
	C ²	Can subscribe for COV from OctetString objects
	C ²	Can subscribe for COV from Positive Integer objects
	C ²	Can subscribe for COV from Time objects
	C ²	Can subscribe for COV from Pulse Converter objects
	C ²	Can subscribe for COV from Access Door objects
	C ²	Can subscribe for COV from Load Control objects
	C ²	Can subscribe for COV from Access Point objects
	C ²	Can subscribe for COV from Credential Data Input objects
	C ²	Can subscribe for COV from Lighting Output objects
	C ²	Can subscribe for COV from Binary Lighting Output objects
	C ²	Can subscribe for COV from Staging objects
	C ²	Can subscribe for COV from Other Standard Object Types
	BTL-C ³	Can cancel subscriptions
	O	Can subscribe for COV from proprietary objects
	N	Can request infinite subscriptions
¹ At least one of these options is required in order to claim conformance to this BIBB. ² At least one of these options is required in order to claim conformance to this BIBB. ³ Support for this option is suggested except in the case where the device is able to generate infinite subscriptions in which case it is required.		
Data Sharing - Change Of Value - B		
	R	Base Requirements
	R	Supports Lifetimes up to 8 hours in duration
	R ²	Supports 5 concurrent COV subscribers
	C ¹	Supports COV for Analog Input objects
	C ¹	Supports COV for Analog Output objects

	C ¹	Supports COV for Analog Value objects
	C ¹	Supports COV for Binary Input objects
	C ¹	Supports COV for Binary Output objects
	C ¹	Supports COV for Binary Value objects
	C ¹	Supports COV for Life Safety Point objects
	C ¹	Supports COV for Life Safety Zone objects
	C ¹	Supports COV for Loop objects
	C ¹	Supports COV for Multi-state Input objects
	C ¹	Supports COV for Multi-state Output objects
	C ¹	Supports COV for Multi-state Value objects
	C ¹	Supports COV for CharacterString Value objects
	C ¹	Supports COV for Date Value objects
	C ¹	Supports COV for Date Pattern Value objects
	C ¹	Supports COV for DateTime Value objects
	C ¹	Supports COV for DateTime Pattern Value objects
	C ¹	Supports COV for Integer Value objects
	C ¹	Supports COV for Large Analog Value objects
	C ¹	Supports COV for Positive Integer Value objects
	C ¹	Supports COV for Time Value objects
	C ¹	Supports COV for Time Pattern Value objects
	C ¹	Supports COV for OctetString Value objects
	C ¹	Supports COV for Pulse Converter objects
	C ¹	Supports COV for Access Door objects
	C ¹	Supports COV for Load Control objects
	C ¹	Supports COV for Access Point objects
	C ¹	Supports COV for Credential Data Input objects
	C ¹	Supports COV for Lighting Output objects
	C ¹	Supports COV for Binary Lighting Output objects
	C ¹	Supports COV for Staging objects
	C ¹	Supports COV for Other Standard Object Types
	O	Supports COV for proprietary objects
	S	Will accept infinite COV subscriptions
¹ At least one of these options is required in order to claim conformance to this BIBB.		
² BTL-R if the IUT claims a revision before Protocol Revision 4.		

Test Plan Changes

[Add 4.9.X]

4.9.X Can Subscribe for COV from Other Standard Object Types

The IUT can subscribe for, receive, and process Change of Value notifications from other standard object types.

135.1-2023 - 9.2.1.1 - Change of Value Notifications		
	Test Conditionality	This test can be skipped if 9.3.2.1 is executed.
	Test Directives	Execute this test with each standard object type not specified in 135-2020 Table 13-1 that supports COV
	Testing Hints	Standard object types could include Calendar or Schedule objects.
135.1-2023 - 9.3.2.1 - Change of Value Notifications		
	Test Conditionality	This test can be skipped if 9.2.1.1 is executed.
	Test Directives	Test one instance of each object type.
	Testing Hints	Standard object types could include Calendar or Schedule objects.

[Add 4.10.X]

4.10.X Supports COV for Other Standard Object Types

The IUT accepts COV subscriptions and initiate COV notifications for other standard objects.

BTL - 8.2.X2 - Change of Value Notification from Other Standard Object Types		
	Test Conditionality	This test may be skipped if 8.3.X2 is executed.
	Test Directives	Execute this test with each standard object type not specified in 135-2020 Table 13-1 that supports COV.
	Testing Hints	
BTL - 8.3.X2 - Change of Value Notification from Other Standard Object Types		
	Test Conditionality	This test may be skipped if 8.2.X2 is executed.
	Test Directives	Execute this test with each standard object type not specified in 135-2020 Table 13-1 that supports COV.
	Testing Hints	

Specified Test Changes

[Add 8.2.X2]

8.2.X2 Change of Value Notification from Other Standard Object Types

Purpose: To verify that the IUT can initiate ConfirmedCOVNotification service requests conveying a change of value for an object (O1) not listed in 135-2020 Table 13-1.

Test Concept: A subscription for COV notifications is established, using a Lifetime of L. L shall be set to a value less than 24 hours and large enough to complete the test. The value of the Present_Value is changed, and it is verified that a COV notification is received. If the Status_Flags is present and can be made to change, it is verified that when Status_Flags changes a COV notification is received.

Configuration Requirements: None.

Test Steps:

1. TRANSMIT SubscribeCOV-Request,
 'Subscriber Process Identifier' = (P1, any value > 0 chosen by the TD),
 'Monitored Object Identifier' = O1,
 'Issue Confirmed Notifications' = TRUE,
 'Lifetime' = L
2. RECEIVE BACnet-SimpleACK-PDU
3. BEFORE **Notification Fail Time**
4. RECEIVE ConfirmedCOVNotification-Request,
 'Subscriber Process Identifier' = (P1),
 'Initiating Device Identifier' = IUT,
 'Monitored Object Identifier' = O1,
 'Time Remaining' = (any value appropriate for the Lifetime selected),
 'List of Values' = (the initial values for Present_Value and Status_Flags (if O1 supports Status_Flags))
5. TRANSMIT BACnet-SimpleACK-PDU
6. MAKE (Present_Value change)
7. BEFORE **Notification Fail Time**
8. RECEIVE ConfirmedCOVNotification-Request,
 'Subscriber Process Identifier' = (P1),
 'Initiating Device Identifier' = IUT,
 'Monitored Object Identifier' = O1,
 'Time Remaining' = (any value appropriate for the Lifetime selected),
 'List of Values' = (updated values for Present_Value and Status_Flags (if O1 supports Status_Flags))
9. IF (Status_Flags are present and can be changed by some action) THEN {
10. MAKE (Status_Flags change)
11. BEFORE **Notification Fail Time**
12. RECEIVE ConfirmedCOVNotification-Request,
 'Subscriber Process Identifier' = (P1),

```
        'Initiating Device Identifier' = IUT,  
        'Monitored Object Identifier' = O1,  
        'Time Remaining' = (any value appropriate for the Lifetime selected),  
        'List of Values' = (updated values for Present_Value, Status_Flags)  
13.    TRANSMIT BACnet-SimpleACK-PDU  
    }
```

[Add 8.3.X2]

8.3.X2 Change of Value Notification from Other Standard Object Types

Purpose: To verify that the IUT can initiate UnconfirmedCOVNotification service requests conveying a change of value for an object (O1) not listed in 135-2020 Table 13-1.

Test Steps: The steps for this test case are identical to the test steps in 8.2.X2 except that the SubscribeCOV service request in step 1 shall have a value of FALSE for the 'Issue Confirmed Notifications' parameter, all of the ConfirmedCOVNotification requests shall be UnconfirmedCOVNotification requests, and there is no acknowledgment of the unconfirmed services. The MAC address used for the notification message shall be such that the TD is one of the recipients.

BTL-23.3 imp3-15: Timestamp format for test 9.1.2.1 [BTLWG-1524]

Overview:

Addendum 135-2016br (135-2016br-4), PR21, deprecated the ‘time’ form of the BACnetTimeStamp datatype because a time value without a date is too ambiguous. Test 9.1.2.1 explicitly asks for the ‘Time’ format in the time of acknowledgement parameter, so we need to fix that. There was also an error identified in Step 8 regarding the “Time Stamp” value to use in the test.

Changes:

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, change all references for test 9.1.2.1 from 135.1-2023 to BTL]

Specified Test Changes

[Move test 9.1.2.1 to BTL Specified Tests and modify]

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

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

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

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

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

Test Steps:

1. MAKE (a change that triggers the detection of an alarm event in the IUT)
2. WAIT (D1)
3. BEFORE **Notification Fail Time**
4. RECEIVE ConfirmedEventNotification-Request,
 - 'Process Identifier' = (PID: the process identifier configured for this event),
 - 'Initiating Device Identifier' = IUT,
 - 'Event Object Identifier' = (OI: the object detecting the alarm),
 - 'Time Stamp' = (T1: any valid time stamp),
 - 'Notification Class' = (NC: the notification class configured for this event),
 - 'Priority' = (PI: the priority configured for this event type),
 - 'Event Type' = (ET: any valid event type),

'Message Text' = (MT: optional, any valid message text),
 'Notify Type' = (NT: the notify type configured for the event),
 'AckRequired' = TRUE,
 'From State' = (S1: any appropriate event state),
 'To State' = (S1/S2: any appropriate event state),
 'Event Values' = (the values appropriate to the event type)

45. TRANSMIT BACnet-SimpleACK-PDU

56. RECEIVE

DESTINATION = (a device other than the TD),

SOURCE = IUT,

ConfirmedEventNotification-Request,

'Process Identifier' = (PID: the process identifier configured for this event),
 'Initiating Device Identifier' = IUT,
 'Event Object Identifier' = (OI: the object detecting the alarm),
 'Time Stamp' = (T1),
 'Notification Class' = (NC: the notification class configured for this event),
 'Priority' = (P: the priority configured for this event type),
 'Event Type' = (E1/E2: any valid event type),
 'Message Text' = (MT: optional, any valid message text),
 'Notify Type' = (NT: the notify type configured for the event),
 'AckRequired' = TRUE,
 'From State' = (S1: any appropriate event state),
 'To State' = (S2: any appropriate event state),
 'Event Values' = (the values appropriate to the event type)

67. TRANSMIT BACnet-SimpleACK-PDU

78. VERIFY (the 'Event Object Identifier' from the event notification) OI,

Acked_Transitions = (appropriate bit FALSE, the others TRUE)

89. TRANSMIT AcknowledgeAlarm-Request,

'Acknowledging Process Identifier' = (P1: the value of the 'Process Identifier' parameter in the event notification),
 'Event Object Identifier' = (OI: the 'Event Object Identifier' from the event notification),
 'Event State Acknowledged' = (S2: the state specified in the 'To State' parameter of the notification),
 'Time Stamp' = (any valid time stamp older than T1),
 'Acknowledgment Source' = (any valid value)
 'Time of Acknowledgment' = (the current time using a Time format)

910. RECEIVE BACnet-Error-PDU

Error Class = SERVICES,

Error Code = INVALID TIME STAMP

1011. VERIFY (the 'Event Object Identifier' from the event notification) OI,

Acked_Transitions = (appropriate bit FALSE, the others TRUE)

1112. TRANSMIT AcknowledgeAlarm-Request,

'Acknowledging Process Identifier' = (PID: the process identifier configured for this event),
 'Event Object Identifier' = (EI: the 'Event Object Identifier' from the event notification),
 'Event State Acknowledged' = (S2: the state specified in the 'To State' parameter of the notification),
 'Time Stamp' = T1,
 'Time of Acknowledgment' = (the current time using a Time format)

1213. RECEIVE BACnet-SimpleACK-PDU

13. IF (Protocol_Revision is present AND Protocol_Revision >= 1) THEN

14. BEFORE Notification Fail Time

1415. RECEIVE

ConfirmedEventNotification-Request,

'Process Identifier' = (PID: the process identifier configured for this event),
 'Initiating Device Identifier' = IUT,
 'Event Object Identifier' = (OI: the object detecting the alarm),
 'Time Stamp' = (T2: any valid time stamp),
 'Notification Class' = (NC: the notification class configured for this event),
 'Priority' = (P: the priority configured for this event type),
 'Event Type' = (E1: any valid event type),
 'Message Text' = (optional, any valid message text),
 'Notify Type' = ACK_NOTIFICATION,
 'To State' = (S1 or S2)

~~ELSE~~~~BEFORE Notification Fail Time~~~~RECEIVE~~~~ConfirmedEventNotification Request,~~~~'Process Identifier' = (the process identifier configured for this event);~~~~'Initiating Device Identifier' = IUT;~~~~'Event Object Identifier' = (O1 the object detecting the alarm);~~~~'Time Stamp' = (T2: any valid time stamp);~~~~'Notification Class' = (the notification class configured for this event);~~~~'Priority' = (the priority configured for this event type);~~~~'Event Type' = (E1 any valid event type);~~~~'Message Text' = (optional, any valid message text);~~~~'Notify Type' = ACK_NOTIFICATION~~~~14/16. TRANSMIT BACnet-SimpleACK-PDU~~~~15. IF (Protocol_Revision is present AND Protocol_Revision >= 1) THEN~~~~17. RECEIVE~~~~DESTINATION = (a device other than the TD),~~~~SOURCE = IUT,~~~~ConfirmedEventNotification-Request,~~~~'Process Identifier' = (PID the process identifier configured for this event),~~~~'Initiating Device Identifier' = IUT,~~~~'Event Object Identifier' = (O1 the object detecting the alarm),~~~~'Time Stamp' = (T2),~~~~'Notification Class' = (NC the notification class configured for this event),~~~~'Priority' = (P1 the priority configured for this event type),~~~~'Event Type' = (E1 any valid event type),~~~~'Message Text' = (optional, any valid message text),~~~~'Notify Type' = ACK_NOTIFICATION,~~~~'To State' = (S1 or S2)~~~~ELSE~~~~RECEIVE~~~~DESTINATION = (a device other than the TD),~~~~SOURCE = IUT,~~~~ConfirmedEventNotification Request,~~~~'Process Identifier' = (the process identifier configured for this event);~~~~'Initiating Device Identifier' = IUT;~~~~'Event Object Identifier' = (O1 the object detecting the alarm);~~~~'Time Stamp' = (T2);~~~~'Notification Class' = (NC the notification class configured for this event);~~~~'Priority' = (the priority configured for this event type);~~~~'Event Type' = (E1 any valid event type);~~~~'Message Text' = (optional, any valid message text);~~~~'Notify Type' = ACK_NOTIFICATION~~~~16/18. TRANSMIT BACnet-SimpleACK-PDU~~~~17/19. VERIFY (the 'Event Object Identifier' from the event notification) O1, Acked_Transitions = (TRUE,TRUE,TRUE)~~

BTL-23.3 imp3-16: Add DS-COV-B Tests for the Load Control Object [BTLWG-1544]

Overview:

Update Test Plan 4.10.29 for Load Control object and add appropriate tests.

Changes:

Checklist Changes

None

Test Plan Changes

[Change Clause 4.10.29]

4.10.29 Supports COV for Load Control Objects

The IUT accepts COV subscriptions and initiates COV notifications for Load Control objects.

BTL - 8.2.X1 - Change of Value Notification from Load Control Object (ConfirmedCOVNotification)		
	Test Conditionality	This may be skipped if 8.3.X1 is executed.
	Test Directives	
	Testing Hints	
BTL - 8.3.X1 - Change of Value Notification from Load Control Object (UnconfirmedCOVNotification)		
	Test Conditionality	This may be skipped if 8.2.X1 is executed.
	Test Directives	
	Testing Hints	

Specified Test Changes

[Add new test 8.2.X1]

8.2.X1 Change of Value Notification from a Load Control Object (ConfirmedCOVNotification)

Purpose: To verify that the IUT can initiate ConfirmedCOVNotification service requests conveying a change of value for a Load Control object (O1) when the properties specified in the standard changes.

Test Concept: A subscription for COV notifications is established, using a Lifetime of L. L shall be set to a value less than 24 hours and large enough to complete the test. The value of each property is changed and it is verified that a COV notification is received.

Configuration Requirements: None.

Test Steps:

1. TRANSMIT SubscribeCOV-Request,
 'Subscriber Process Identifier' = (P1, any value > 0 chosen by the TD),
 'Monitored Object Identifier' = O1,
 'Issue Confirmed Notifications' = TRUE,
 'Lifetime' = L
2. RECEIVE BACnet-SimpleACK-PDU
3. BEFORE Notification Fail Time
4. RECEIVE ConfirmedCOVNotification-Request,
 'Subscriber Process Identifier' = (P1),
 'Initiating Device Identifier' = IUT,

```

        'Monitored Object Identifier' = O1,
        'Time Remaining' = (any value appropriate for the Lifetime selected),
        'List of Values' = (the initial values for Present_Value, Status_Flags, Requested_Shed_Level,
                           Start_Time, Shed_Duration, and Duty_Window)
5. TRANSMIT BACnet-SimpleACK-PDU
6. MAKE (Present_Value change)
7. BEFORE Notification Fail Time
8. RECEIVE ConfirmedCOVNotification-Request,
        'Subscriber Process Identifier' = (P1),
        'Initiating Device Identifier' = IUT,
        'Monitored Object Identifier' = O1,
        'Time Remaining' = (any value appropriate for the Lifetime selected),
        'List of Values' = (updated values for Present_Value, Status_Flags, Requested_Shed_Level,
                           Start_Time, Shed_Duration, and Duty_Window)
9. IF (Status_Flags can be changed by some action) THEN {
10. MAKE (Status_Flags change)
11. BEFORE Notification Fail Time
12. RECEIVE ConfirmedCOVNotification-Request,
        'Subscriber Process Identifier' = (P1),
        'Initiating Device Identifier' = IUT,
        'Monitored Object Identifier' = O1,
        'Time Remaining' = (any value appropriate for the Lifetime selected),
        'List of Values' = (updated values for Present_Value, Status_Flags, Requested_Shed_Level,
                           Start_Time, Shed_Duration, and Duty_Window)
13. TRANSMIT BACnet-SimpleACK-PDU
    }
14. REPEAT PROP1 = (Request_Shed_Level, Start_Time, Shed_Duration, Duty_Window) DO {
15. WRITE O1, PROP1 = (any value that differs from the value of PROP1 in the last COV Notification)
16. BEFORE Notification Fail Time
17. RECEIVE ConfirmedCOVNotification-Request,
        'Subscriber Process Identifier' = (P1),
        'Initiating Device Identifier' = IUT,
        'Monitored Object Identifier' = O1,
        'Time Remaining' = (any value appropriate for the Lifetime selected),
        'List of Values' = (updated values for Present_Value, Status_Flags, Requested_Shed_Level,
                           Start_Time, Shed_Duration, and Duty_Window)
    }

```

[Add new test 8.3.X1]

8.3.X1 Change of Value Notification from a Load Control Object (UnconfirmedCOVNotification)

Purpose: To verify that the IUT can initiate UnconfirmedCOVNotification service requests conveying a change of the value for a Load Control object when the properties specified in the standard change.

Test Steps: The steps for this test case are identical to the test steps in 8.2.X1 except that the SubscribeCOV service request in step 1 shall have a value of FALSE for the 'Issue Confirmed Notifications' parameter, all of the ConfirmedCOVNotification requests shall be UnconfirmedCOVNotification requests, and there is no acknowledgment of the unconfirmed services. The MAC address used for the notification message shall be such that the TD is one of the recipients.

BTL-23.3 imp3-17: Fix Test Plan Descriptions in DS-COV-B [BTLWG-1545]

Overview:

The description in many of the clauses in 4.10 (DS-COV-B) are copied form 4.9 (DS-COV-A).

Changes:

Checklist Changes

None

Test Plan Changes

[Change Clauses 4.10.4 to 4.10.34 except 4.10.29]

Change the description in Clauses 4.10.4 to 4.10.34 except 4.10.29 from:

"The IUT can subscribe for, receive, and process Change of Value notifications from [object type] objects."

To

"The IUT accepts COV subscriptions and initiates COV notifications for [object type] objects."

Specified Test Changes

None

BTL-23.3 imp3-18: DM-OCD-A Change to Allow any Version Client to Create New Objects [BTLWG-636]**Overview:**

A client is not forbidden from creating or deleting objects at a newer Protocol Revision than the client's Protocol Revision.

Changes:**Checklist Changes**

[Remove the usages footnotes that reference a Protocol_Revision X or higher must be claimed and remove the now unused footnotes from the bottom of the table.]

Device Management - Object Creation and Deletion - A		
	R	Base Requirements
	BTL-R	Can create objects using Object Identifier with no initial values
	S ¹	Can create objects using Object Type with no initial values
	S	Can create objects by Object Identifier with initial values which includes Object Name
	S ¹	Can create objects by Object Type with initial values which includes Object Name
	C ²	Can create and delete Accumulator objects
	C ²	Can create and delete Analog Input objects
	C ²	Can create and delete Analog Output objects
	C ²	Can create and delete Analog Value objects
	C ²	Can create and delete Averaging objects
	C ²	Can create and delete Binary Input objects
	C ²	Can create and delete Binary Output objects
	C ²	Can create and delete Binary Value objects
	C ²	Can create and delete Calendar objects
	C ²	Can create and delete Command objects
	C ²	Can create and delete Event Enrollment objects
	C ^{2,3}	Can create and delete File objects
	C ²	Can create and delete Group objects
	C ²	Can create and delete Life Safety Point objects
	C ²	Can create and delete Life Safety Zone objects
	C ²	Can create and delete Loop objects
	C ²	Can create and delete Multi State Input objects
	C ²	Can create and delete Multi State Output objects
	C ²	Can create and delete Multi State Value objects
	C ²	Can create and delete Notification Class objects
	C ²	Can create and delete Program objects
	C ²	Can create and delete Pulse Converter objects
	C ²	Can create and delete Schedule objects
	C ²	Can create and delete Trend Log objects
	C ²	Can create and delete Structured View objects
	C ²	Can create and delete Load Control objects
	C ²	Can create and delete Access Door objects
	C ²	Can create and delete Proprietary objects
	C ²	Can create and delete Event Log objects
	C ²	Can create and delete Trend Log Multiple objects
	C ²	Can create and delete CharacterString Value objects
	C ²	Can create and delete DateTime Value objects
	C ²	Can create and delete Large Analog Value objects
	C ²	Can create and delete BitString Value objects
	C ²	Can create and delete OctetString Value objects
	C ²	Can create and delete Time Value objects
	C ²	Can create and delete Integer Value objects

	C ²	Can create and delete Positive Integer Value objects
	C ²	Can create and delete Date Value objects
	C ²	Can create and delete DateTime Pattern Value objects
	C ²	Can create and delete Time Pattern Value objects
	C ²	Can create and delete Date Pattern Value objects
	C ²	Can create and delete Network Security objects
	C ²	Can create and delete Global Group objects
	C ²	Can create and delete Access Point objects
	C ²	Can create and delete Access Zone objects
	C ²	Can create and delete Access User objects
	C ²	Can create and delete Access Rights objects
	C ²	Can create and delete Access Credential objects
	C ²	Can create and delete Credential Data objects
	C ²	Can create and delete Notification Forwarder objects
	C ²	Can create and delete Alert Enrollment objects
	C ²	Can create and delete Channel objects
	C ²	Can create and delete Lighting Output objects
	C ²	Can create and delete Binary Lighting Output objects
	C ²	Can create and delete Network Port objects
	C ²	Can create and delete Timer objects
	C ²	Can create and delete Elevator Group objects
	C ²	Can create and delete Lift objects
	C ²	Can create and delete Escalator objects
	C ²	Can create and delete Staging objects
	C ²	Can create and delete Audit Reporter objects
	C ²	Can create and delete Audit Log objects
¹ For each object type where the IUT supports creation with the CreateObject service, the Object_Identifier form of CreateObject shall be supported. ² At least one of these is required in order to claim conformance to this BIBB. ³ IUT should not claim this functionality if the only method of creating a File object is by initiating the device restore procedure. ⁴ Protocol Revision 16 or higher must be claimed. ⁵ Protocol Revision 16 or higher must be claimed. ⁶ Protocol Revision 16 or higher must be claimed. ⁷ Protocol Revision 16 or higher must be claimed.		

Test Plan Changes

None

Specified Test Changes

None

BTL-23.3 imp3-19: DM-R-B and DS-COVU-B Clarification [BTLWG-1559]

Overview:

It is not clear that DS-COVU-B is required to be claimed when the IUT claims DM-R-B. The standard also says the process identifier shall be 0 when issuing a restart notification however the test says 'should'.

13.1.1 Unsubscribed COV Notifications

Some objects may share information by generating UnconfirmedCOVNotification messages without using COV subscriptions. As described in Clause 13.7, such notifications set the Subscriber Process Identifier parameter to zero to identify them as unsubscribed.

Changes:

Checklist Changes

None

Test Plan Changes

[In the Test Plan section, deletions should be shown in ~~strikethrough~~, and additions in *italics*
[If a complete new section is being added in, do not use italics]

[Test Plan 4.18.1 - DS-COVU-B - Change the Test Conditionality to allow the test to be if only supported via restart COV.]

4.18.1 Base Requirements

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

BTL - 8.3.9 - Unsubscribed Change of Value Notifications		
	Test Conditionality	Must be executed, unless the IUT only supports unsubscribed COVU through restart notifications.
	Test Directives	
	Testing Hints	

[Test Plan 8.20.1 DM-Restart-B- Add verify checklist with test directives to Verify that the IUT claims support for DS-COVU-B]

8.20.1 Base Requirements

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

135.1-2023 - 8.3.10 - Device Restart Notifications		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	Repeat the test with unicast and broadcast recipients. Repeat the test with each of the restart methods that the device supports and which can be performed at will (warm start, cold start, power cycle, power lost, etc).
Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verity that the IUT claims support for DS-COVU-B.
	Testing Hints	1.

Specified Test Changes

[Modify existing BTL Specified Test 8.3.9 as shown]

8.3.9 Unsubscribed Change of Value Notifications

Reason for Change: Add Process ID Requirement and Abort Conditionality to test.

Unsubscribed COV notifications differ from subscribed COV notifications that use the UnconfirmedCOVNotification service in two respects. First, no subscription is required. Second, the 'Subscriber Process Identifier' parameter ~~usually has~~ **shall have** a value of zero.

Purpose: To verify that the IUT can initiate UnconfirmedCOVNotification service requests when no subscription for the COV notification has been made.

Test Concept: The IUT is configured to send unsubscribed COV notifications. The TD then waits for the notification. Given that there is no defined trigger, the vendor shall inform the tester how to make the IUT generate the notifications if they are not sent periodically.

Test Steps:

1. MAKE (the IUT send an unsubscribed COV notification)

2. BEFORE **Notification Fail Time**

RECEIVE UnconfirmedCOVNotification-Request,

'Subscriber Process Identifier' = ~~(any valid process ID)~~0,

'Initiating Device Identifier' = IUT,

'Monitored Object Identifier' = (any valid object identifier),

'Time Remaining' = 0,

'List of Values' = (any valid properties and values from the monitored object)

BTL-23.3 imp3-20: B_SC Must Support Time Synchronization [BTLWG-1564]

Overview:

A B/SC device must maintain accurate time. See Clause AB.7.4 and AB.7.5.1. Addendum fix1 added entries to the Test Plan and a new test. This proposal adds Checklist items and cleans up the Test Plan to make this requirement more visible to the customer when filling out the Checklist.

Changes:

Checklist Changes

9 Data Link Layer

[Modify Section 9 – Data Link Layer]

Support	Listing	Option
...		
Data Link Layer - Secure Connect		
	R	Base Requirements
	C ¹	Is able to operate as a node without a local hub function
	C ¹	Is able to operate as a hub
	O	Supports direct connections
	O ²	Is able to accept direct connections
	O ²	Is able to initiate direct connections
	O	Supports configuration through Network Port object
	C ^{3,4}	Supports Data Attributes as of Protocol_Revision 25
	C ⁵	Supports DM-TS-B
	C ⁵	Supports DM-UTC-B
	C ⁵	Supports Time Synchronization by Some Other Method
<div>¹ At least one of these options must be supported.</div> <div>² At least one of these options must be supported if the device supports direct connections.</div> <div>³ Required if the IUT claims Protocol_Revision 25 or higher.</div> <div>⁴ Contact BTL for interim tests for this functionality.</div> <div>⁵ At least one of these options must be supported</div>		
...		

Test Plan Changes

[Do not include changes to Clause 9.9.1 from Addendum fix1]

9.9.1 Base Requirements

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

...		
Verify Checklist		
	Test Conditionality	Must be executed unless 12.5.X.1 is executed.
	Test Directives	Verify that the IUT claims support for DM TS B and/or DM UTC B.
	Testing Hints	

BTL - 12.5.X.1 - Verify Time Synchronization Test		
	Test Conditionality	Must be executed unless the IUT claims support for DM-TS-B and/or DM-UTC-B.
	Test Directives	
	Testing Hints	

[Add Clause 9.9.X1]

9.9.X1 Supports DM-TS-B

The IUT supports DM-TS-B.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DM-TS-B.
	Testing Hints	

[Add Clause 9.9.X2]

9.9.X2 Supports DM-UTC-B

The IUT supports DM-UTC-B.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DM-UTC-B.
	Testing Hints	

[Add Clause 9.9.X3]

9.9.X3 Supports Time Synchronization by Some Other Method

The IUT supports time synchronization by some other method.

BTL - 12.5.X.1 - Verify Time Synchronization Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

Specified Test Changes

None

BTL-23.3 imp3-21: Do Not Allow Absent DVA for Broadcast [BTLWG-1566]**Overview:**

12.5.2.1.2 step 1, should not allow DVA to be absent for a broadcast message sent to the hub. Not unicast so must be broadcast VMAC. See AB.2.1.

Changes:

Checklist Changes

None

Test Plan Changes

None

Specified Test Changes

[Change BTL Specified Tests]

12.5.2.1.2 Local Broadcast Execution Test

Reason for Change: Allow for an invalid Destination Virtual Address. Disallowed for an absent DVA for a broadcast.

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'X'FFFFFFFFFFFFFFF, -- the local broadcast VMAC)
 -- 'Destination Options' absent
 'Data Options' = ({X'41'}), -- Secure Path
 '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'FFFFFFFF'X'FFFFFFFFFFFFFFF, -- 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 D4's VMAC or X'FFFFFFF'X'FFFFFFFFFFFFFFF, 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-23.3 imp3-22: Register Foreign Device Tests when NPO Supported [BTLWG-1444]

Overview:

The task focuses on updating three specific tests (12.3.8.2, 12.3.8.4, and 12.3.8.6) that deal with "Register-Foreign-Device" parameters. Originally, these tests relied on end-user interfaces for configuration. However, since Protocol Revision (PR) 17, configuration should utilize Network Port Objects (NPOs).

Related test: 7.3.2.46.3.3.1 RENEW_FD_REGISTRATION Command Test

Changes:

Checklist Changes
[None]

Test Plan Changes
[In BTL Test Plan, modify tests 12.3.8.2, 12.3.8.4, and 12.3.8.6 under section 9.3.3] [In BTL Test Plan, add new test 12.3.8.X under section 9.3.3]

9.3.3 Is Able to Operate in Foreign Mode

135.1-2023 - 12.3.8.1 - Registering as a Foreign Device	
Test Conditionality	Must be executed if the IUT claims Protocol Revision < 17.
Test Directives	Repeat this test with the IUT configured to register as a foreign device with the TD that is using a valid host.ip-address and again with a valid host.name.
Testing Hints	
135.1-2023 - 12.3.8.2 - Register-Foreign-Device Enable and Disable Test	
Test Conditionality	Must be executed if the IUT claims Protocol Revision < 17.
Test Directives	
Testing Hints	
135.1-2023 - 12.3.8.4 - BBMD Address Configuration Test	
Test Conditionality	Must be executed if the IUT claims Protocol Revision < 17.
Test Directives	
Testing Hints	
135.1-2023 - 12.3.8.6 - Time-to-Live Configuration Test	
Test Conditionality	Must be executed if the IUT claims Protocol Revision < 17 and Time-to-Live is configurable.
Test Directives	
Testing Hints	
BTL - 12.3.8.X1 - Register-Foreign-Device when NPOs Supported	
Test Conditionality	Must be executed if the IUT claims Protocol Revision >= 17.
Test Directives	Repeat this test with the IUT configured to register as a foreign device with the TD that is using a valid BBMD Address, once using the host ip-address and again using the host name.
Testing Hints	

9.8.3 Is Able to Operate in Foreign Mode

BTL - 12.3.8.X1 - Register-Foreign-Device when NPOs Supported	
Test Conditionality	Must be executed.
Test Directives	Repeat this test with the IUT configured to register as a foreign device with the TD that is using a valid BBMD Address, once using the host ip-address and again using the host name.
Testing Hints	

Specified Test Changes

[Add new test to BTL Specified Tests]

12.3.8.X1 Register-Foreign-Device when NPOs Supported

Reason for Change: No test for this configuration.

Purpose: To validate whether the Network Port Object can configure the dispatch and parameters of Register-Foreign-Device requests.

Test Concept: To enable and disable Register-Foreign-Device requests use the BACnet_IP_Mode property and configure the 'BBMD Address' and 'Time-to-Live' parameters through the FD_BBMD_Address and FD_Subscription_Lifetime properties in the Network Port Object.

Configuration Requirements: BBMD1 is the TD simulating a correctly functioning BBMD implementation. The IUT's Network Port object is initially configured for BACnet/IP or BACnet/IPv6 in NORMAL mode. Mode represents BACnet_IP_Mode for BACnet/IP and BACnet_IPv6_Mode for BACnet_IPv6.

The Network Port object shall have no pending changes.

Test Steps:

- ```
-- make sure our initial conditions are good
1. VERIFY Changes_Pending = FALSE
2. VERIFY Reliability = NO_FAULT_DETECTED
3. VERIFY Mode = NORMAL

-- update Mode, BBMD address and subscription lifetime
4. IF (Mode is writable) THEN
5. WRITE Mode = FOREIGN
 ELSE
6. MAKE (Mode = FOREIGN)
7. WRITE FD_BBMD_Address = BBMD1
8. WRITE FD_Subscription_Lifetime = T1 (arbitrary value in seconds)
9. VERIFY Changes_Pending = TRUE
10. TRANSMIT ReinitializeDevice-Request,
 'Reinitialized State of Device' = WARMSTART | ACTIVATE_CHANGES,
 'Password' = (any valid password)
11. RECEIVE BACnet-SimpleACK-PDU
12. WAIT Activate Changes Fail Time
13. VERIFY Mode = FOREIGN
14. VERIFY FD_BBMD_Address = BBMD1
15. VERIFY FD_Subscription_Lifetime = T1

-- verify Register-Foreign-Device request in TD
16. WAIT (T1 seconds)
17. RECEIVE DA = BBMD1,
 Register-Foreign-Device,
 'Time-to-Live' = T1
18. TRANSMIT BVLC-Result,
 'Result Code' = Successful completion

-- verify that the Register-Foreign-Device requests can be disabled
19. VERIFY Changes_Pending = FALSE
20. IF (Mode is writable) THEN
21. WRITE Mode = NORMAL
22. VERIFY Changes_Pending = TRUE
23. TRANSMIT ReinitializeDevice-Request,
 'Reinitialized State of Device' = WARMSTART | ACTIVATE_CHANGES,
 'Password' = (any valid password)
```

24. RECEIVE BACnet-SimpleACK-PDU
25. WAIT Activate Changes Fail Time
- ELSE
26. MAKE (the IUT enter NORMAL mode)
27. VERIFY Mode = NORMAL
28. WAIT (more than T1 seconds)
29. CHECK (that the IUT did not send any Register-Foreign-Device requests)