BACnet® TESTING LABORATORIES ADDENDA

Addendum Imp3 to BTL Test Package 26.1

Revision final
Revised 6/29/2026

Approved by the BTL Working Group on April 27, 2026;
Approved by the BTL Working Group Voting Members on June 22, 2026;
Published on July 3, 2026.

[This foreword and the "Overview" on the following pages are not part of this Test Package. They are merely informative and do not contain requirements necessary for conformance to the Test Package.]

FOREWORD

The purpose of this addendum is to present current changes being made to the BTL Test Package. These modifications are the result of change proposals made pursuant to the continuous maintenance procedures and of deliberations within the BTL-WG Committee. The changes are summarized below.

In the following document, language to be added to existing clauses within the BTL Test Package 26.1 is indicated through the use of *italics*, while deletions are indicated by ~~strikethrough~~. Where entirely new subclauses are proposed to be added, plain type is used throughout.

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

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

BTL-26.1 imp3-1: Fix Passback-Mode Test [BTLWG-1619]

Overview:

135.1-2025 - 7.3.2.42.4 Passback Mode test, step 9 states the Access_Event = GRANTED when the value should be PASSBACK_DETECTED when the Passback_Mode = SOFT_PASSBACK. See Clause 12.32.21.

Changes:

Checklist Changes

None

Test Plan Changes

3.45 Access Zone Object

[Update Section 3.45.5 to remove old test and include new tests]

3.45.5 Supports Passback_Mode Property

The IUT supports Passback_Mode property.

~~**135.1-2025 - 7.3.2.42.4 - Passback Mode Test**~~		
	~~**Test Conditionality**~~	~~If the IUT does not support soft passback or hard passback mode then those tests shall be skipped. If Passback_Timeout is not supported then those steps shall be skipped. If the passback exemption of the Access Credential object type is not supported then those steps shall be skipped.~~
	~~**Test Directives**~~	
	~~**Testing Hints**~~	
BTL - 7.3.2.42.4.X1 - PASSBACK_OFF Mode Test		
	Test Conditionality	*Must be executed.*
	Test Directives	
	Testing Hints	
BTL - 7.3.2.42.4.X2 - SOFT_PASSBACK Mode Test		
	Test Conditionality	*Must be executed.*
	Test Directives	
	Testing Hints	
BTL - 7.3.2.42.4.X3 - HARD_PASSBACK Mode Test		
	Test Conditionality	*Must be executed.*
	Test Directives	
	Testing Hints	
BTL - 7.3.2.42.4.X4 - Passback Exemption Test		
	Test Conditionality	*This test shall be executed if at least one of the IUT's Access Credential objects support the Authorization_Exemption property.*
	Test Directives	
	Testing Hints	

Specified Test Changes

7.3.2.42 Access Zone Object Tests

7.3.2.42.4 Passback Mode ~~Test~~*Tests*

Reason for Change: Replace single Passback Mode Test with smaller individual tests for clarity.

~~Purpose: To verify the passback functionality and passback exemption.~~

~~Test Concept: A valid credential is presented at the entry access point to this access zone. When the credential is presented a second time, a passback notification should be generated, and if the passback mode is set to hard passback, then access to the zone should be denied. If the credential has a passback exemption, then access will never be denied due to a passback violation.~~

~~Configuration Requirements: See Clause 7.3.2.42. This test requires the following additional configuration:~~

- ~~a) The Access Zone shall not be out of service.~~
- ~~b) An access point which is an entry point to this zone shall be represented by Access Point object AP1. The Authoriztion_Mode property of AP1 shall have the value Authorize.~~
- ~~c) An access point which is an exit point to this zone shall be represented by Access Point object AP2. The Authoriztion_Mode property of AP2 shall have the value Authorize.~~
- ~~d) An active credential with valid access rights for the access point AP1 and AP2 shall be represented by Access Credential object C1.~~
- ~~e) The C1.Authorization_Exemptions list shall be empty.~~

~~Test Steps:~~

~~- verify soft passback mode~~
~~1. MAKE (Passback_Mode = SOFT_PASSBACK)~~
~~2. READ EventTag = AP1,Access_Event_Tag~~
~~3. MAKE (present credential C1 at credential reader for access point AP1)~~
~~4. VERIFY AP1, Access_Event = GRANTED~~
~~5. VERIFY AP1,Access_Event_Time = (the time that credential C1 was presented)~~
~~6. VERIFY AP1,Access_Event_Credential = C1~~
~~7. VERIFY AP1,Access_Event_Tag = (EventTag + 1)~~
~~8. MAKE (present credential C1 at credential reader for access point AP1)~~
~~9. VERIFY AP1,Access_Event = GRANTED~~
~~10. VERIFY (AP1.Access_Event_Time = (the time that credential C1 was presented most recently)~~
~~11. VERIFY AP1.Access_Event_Credential = C1~~
~~12. VERIFY AP1,Access_Event_Tag = (EventTag +2)~~
~~13. MAKE (present credential C1 at credential reader for access point AP2)~~

~~- verify hard passback mode~~
~~14. MAKE (Passback_Mode = HARD_PASSBACK)~~
~~15. READ EventTag = AP1,Access_Event_Tag~~
~~16. MAKE (present credential C1 at credential reader for access point AP1)~~
~~17. VERIFY AP1.Access_Event = GRANTED~~
~~18. VERIFY AP1.Access_Event_Time = (the time that credential C1 was presented most recently)~~
~~19. VERIFY AP1.Access_Event_Credential = C1~~
~~20. VERIFY AP1,Access_Event_Tag = EventTag + 1~~
~~21. MAKE (present credential C1 at credential reader for access point AP1)~~
~~22. VERIFY AP1.Access_Event = DENIED_PASSBACK~~
~~23. VERIFY AP1.Access_Event_Time = (the time that credential C1 was presented most recently)~~
~~24. VERIFY AP1.Access_Event_Credential = C1~~
~~25. VERIFY AP1,Access_Event_Tag = (EventTag + 2)~~

~~- verify passback timeout~~
~~26. WAIT (Passback_Timeout)~~
~~27. MAKE (present credential C1 at credential reader for access point AP1)~~
~~28. VERIFY AP1.Access_Event = GRANTED~~
~~29. VERIFY (AP1.Access_Event_Time = the time that credential C1 was presented)~~
~~30. VERIFY (AP1.Access_Event_Credential = C1)~~

~~- verify hard passback off~~
~~31. MAKE (Passback_Mode = PASSBACK_OFF)~~

~~32. READ EventTag = AP1,Access_Event_Tag~~
~~33. MAKE (present credential C1 at credential reader for access point AP1)~~
~~34. VERIFY AP1.Access_Event = GRANTED~~
~~35. VERIFY AP1.Access_Event_Time = (the time that credential C1 was presented most recently)~~
~~36. VERIFY AP1.Access_Event_Credential = C1~~
~~37. VERIFY AP1,Access_Event_Tag = EventTag + 1~~
~~38. MAKE (present credential C1 at credential reader for access point AP1)~~
~~39. VERIFY AP1.Access_Event = GRANTED~~
~~40. VERIFY AP1.Access_Event_Time = (the time that credential C1 was presented most recently)~~
~~41. VERIFY (AP1.Access_Event_Credential = C1~~
~~42. VERIFY AP1,Access_Event_Tag = (EventTag + 2)~~

~~-- verify passback exemption~~
~~43. MAKE (Passback_Mode = HARD_PASSBACK)~~
~~44. MAKE (C1, Authorization_Exemption = (PASSBACK))~~
~~45. READ EventTag = AP1,Access_Event_Tag~~
~~46. MAKE (present credential C1 at credential reader for access point AP1)~~
~~47. VERIFY AP1.Access_Event = GRANTED~~
~~48. VERIFY AP1.Access_Event_Time = (the time that credential C1 was presented most recently)~~
~~49. VERIFY AP1.Access_Event_Credential = C1~~
~~50. VERIFY AP1,Access_Event_Tag = EventTag + 1~~
~~51. MAKE (present credential C1 at credential reader for access point AP1)~~
~~52. VERIFY AP1.Access_Event = GRANTED~~
~~53. VERIFY AP1.Access_Event_Time = (the time that credential C1 was presented most recently)~~
~~54. VERIFY AP1.Access_Event_Credential = C1~~
~~55. VERIFY AP1,Access_Event_Tag = (EventTag + 2)~~

[Change 135.1-2025 - 12.3.11.4]

7.3.2.42.4.X1 PASSBACK_OFF Mode Test

Reason for Change: Add new test for PASSBACK_OFF Mode.

Purpose: To verify the functionality of the Passback_Mode property when set to PASSBACK_OFF mode.

Test Concept: A valid credential is presented at an entry access point of an access zone with Passback_Mode set to PASSBACK_OFF. When the same valid credential is presented again at the same entry access point, without an intervening authorized exit, then a GRANTED Access_Event notification is generated.

Configuration Requirements: See Clause 7.3.2.42. This test requires the following additional configuration:
a) The Access Zone object, AZO1, shall have its Out_Of_Service property set to FALSE.
b) An Access Point object, APO1, representing an entry access point for the Access Zone, shall exist.
c) The Authorization_Mode property of APO1 shall be set to Authorize.
d) A valid credential, VC1, shall exist and shall have access rights for APO1.
e) An Access Credential object, AC1, representing VC1, shall exist.

Test Steps:

1. MAKE (AZO1, Passback_Mode = PASSBACK_OFF)
2. READ EventTag = APO1, Access_Event_Tag
3. MAKE (present credential VC1 at the credential reader for access point APO1)
4. VERIFY APO1, Access_Event = GRANTED
5. VERIFY APO1, Access_Event_Time = (the most recent time credential VC1 was presented.)
6. VERIFY APO1, Access_Event_Credential = AC1
7. VERIFY APO1, Access_Event_Tag = EventTag + 1
8. MAKE (present credential VC1 at the credential reader for access point APO1)
9. VERIFY APO1, Access_Event = GRANTED
10. VERIFY APO1, Access_Event_Time = (the most recent time credential VC1 was presented.)
11. VERIFY APO1, Access_Event_Credential = AC1
12. VERIFY APO1, Access_Event_Tag = (EventTag + 2)

7.3.2.42.4.X2 SOFTPASS_BACK Mode Test

Reason for Change: Add new test for SOFTPASS_BACK Mode.

Purpose: To verify the functionality of the Passback_Mode property when set to SOFT_PASSBACK.

Test Concept: A valid credential is presented at an entry access point of an access zone with Passback_Mode set to SOFT_PASSBACK. When the same valid credential is presented again at the same entry access point, without an intervening authorized exit, a PASSBACK_DETECTED Access_Event notification is generated.

Configuration Requirements: See Clause 7.3.2.42. This test requires the following additional configuration:
a) The Access Zone object, AZO1, shall have its Out_Of_Service property set to FALSE.
b) An Access Point object, APO1, representing an entry access point for the Access Zone, shall exist.
c) The Authorization_Mode property of APO1 shall be set to Authorize.
d) A valid credential, VC1, shall exist and shall have access rights for APO1.
e) An Access Credential object, AC1, representing VC1, shall exist.

Test Steps:
1. MAKE (AZO1, Passback_Mode = SOFT_PASSBACK)
2. READ EventTag = APO1, Access_Event_Tag
3. MAKE (present credential VC1 at the credential reader for access point APO1)
4. VERIFY APO1, Access_Event = GRANTED
5. VERIFY APO1, Access_Event_Time = (the time credential VC1 was presented)
6. VERIFY APO1, Access_Event_Credential = AC1
7. VERIFY APO1, Access_Event_Tag = (EventTag + 1)
8. MAKE (present credential VC1 at the credential reader for access point APO1)
9. VERIFY APO1, Access_Event = PASSBACK_DETECTED
10. VERIFY APO1, Access_Event_Time = (the most recent time credential VC1 was presented.)
11. VERIFY APO1, Access_Event_Credential = AC1
12. VERIFY APO1, Access_Event_Tag = (EventTag +2)
13. IF (Passback_Timeout property is present AND Passback_Timeout > 0) THEN
14. WAIT (Passback_Timeout)
15. MAKE (present credential VC1 at the credential reader for access point APO1)
16. VERIFY APO1, Access_Event = GRANTED
17. VERIFY APO1, Access_Event_Time = (the time credential VC1 was presented)
18. VERIFY APO1, Access_Event_Credential = AC1
19. VERIFY APO1, Access_Event_Tag = (EventTag + 3)

7.3.2.42.4.X3 HARD_PASSBACK Mode Test

Reason for Change: Add new test for HARD_PASSBACK Mode.

Purpose: To verify the functionality of the Passback_Mode property when set to HARD_PASSBACK mode.

Test Concept: A valid credential is presented at an entry access point of an access zone with Passback_Mode set to HARD_PASSBACK. When the same valid credential is presented again at the same entry access point, without an intervening authorized exit, a DENIED_PASSBACK Access_Event notification is generated.

Configuration Requirements: See Clause 7.3.2.42. This test requires the following additional configuration:
a) The Access Zone object, AZO1, shall have its Out_Of_Service property set to FALSE.
b) An Access Point object, APO1, representing an entry access point for the Access Zone, shall exist.
c) The Authorization_Mode property of APO1 shall be set to Authorize.
d) A valid credential, VC1, shall exist and shall have access rights for APO1.
e) An Access Credential object, AC1, representing VC1, shall exist.

Test Steps:
1. MAKE (AZO1, Passback_Mode = HARD_PASSBACK)
2. READ EventTag = APO1, Access_Event_Tag

3. MAKE (present credential VC1 at the credential reader for access point APO1)
4. VERIFY APO1, Access_Event = GRANTED
5. VERIFY APO1, Access_Event_Time = (the most recent time credential VC1 was presented.)
6. VERIFY APO1, Access_Event_Credential = AC1
7. VERIFY APO1, Access_Event_Tag = EventTag + 1
8. MAKE (present credential VC1 at the credential reader for access point APO1)
9. VERIFY APO1, Access_Event = DENIED_PASSBACK
10. VERIFY APO1, Access_Event_Time = (the most recent time credential VC1 was presented.)
11. VERIFY APO1, Access_Event_Credential = AC1
12. VERIFY APO1, Access_Event_Tag = (EventTag + 2)
13. IF (Passback_Timeout property is present AND Passback_Timeout > 0) THEN
14. WAIT (Passback_Timeout)
15. MAKE (present credential VC1 at the credential reader for access point APO1)
16. VERIFY APO1, Access_Event = GRANTED
17. VERIFY APO1, Access_Event_Time = (the time credential VC1 was presented)
18. VERIFY APO1, Access_Event_Credential = AC1
19. VERIFY APO1, Access_Event_Tag = (EventTag + 3)

7.3.2.42.4.X4 Passback Exemption Test

Reason for Change: Add new test to verify Passback Exemption.

Purpose: To verify the functionality of Passback Exemption.

Test Concept: A valid credential with Authorization_Exemption set to PASSBACK, is presented at an entry access point of an access zone with Passback_Mode set to HARD_PASSBACK. When the same valid credential is presented again at the same entry access point, without an intervening authorized exit, a GRANTED Access_Event notification is generated.

Test Conditionality: This test shall be executed if at least one of the IUT's Access Credential objects support the Authorization_Exemption property.

Configuration Requirements: See Clause 7.3.2.42. This test requires the following additional configuration:

a) The Access Zone object, AZO1, shall have its Out_Of_Service property set to FALSE.
b) An Access Point object, APO1, representing an entry access point for the Access Zone, shall exist.
c) The Authorization_Mode property of APO1 shall be set to Authorize.
d) A valid credential, VC1, shall exist and shall have access rights for APO1.
e) An Access Credential object, AC1, representing VC1, shall exist.
f) The Authorization_Exemption property of AC1 shall contain VC1

Test Steps:

1. MAKE (Passback_Mode = HARD_PASSBACK)
2. MAKE (AC1, Authorization_Exemption = PASSBACK)
3. READ EventTag = APO1, Access_Event_Tag
4. MAKE (present credential VC1 at the credential reader for access point APO1)
5. VERIFY APO1, Access_Event = GRANTED
6. VERIFY APO1, Access_Event_Time = (the most recent time credential VC1 was presented.)
7. VERIFY APO1, Access_Event_Credential = AC1
8. VERIFY APO1, Access_Event_Tag = EventTag + 1
9. MAKE (present credential VC1 at the credential reader for access point APO1)
10. VERIFY APO1, Access_Event = GRANTED
11. VERIFY APO1, Access_Event_Time = (the most recent time credential VC1 was presented.)
12. VERIFY APO1, Access_Event_Credential = AC1
13. VERIFY APO1, Access_Event_Tag = (EventTag + 2)

BTL-26.1 imp3-2: Fix Time Count Down Test [BTLWG-1697]

Overview:

Clarify the Note To Tester by simplifying the test.

Changes:

Checklist Changes

None

Test Plan Changes

3.51.1 Base Requirements

~~135.1-2025~~*BTL* - 7.3.2.30.8.1 - Time Count Down Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

Specified Test Changes

[Move test from 135.1-2025 into BTL and modify as shown]

7.3.2.30.8.1 Time Count Down Test

Reason for change: Vendor specified time remaining was not clearly defined. CHECKs were used in the test instead of VERIFY.

Purpose: Verify Subscribed_Recipients entries count Time Remaining down.

Test Concept: Add a subscription to a Notification Forwarder and check that the Subscribed_Recipients Time Remaining value decreases after time has passed.

Note to tester: If the device timing resolution is ~~significantly~~ greater than 1 minute, wait *the time specified by the vendor to cause the time remaining to decrease* ~~the wait times called for by the test steps may be extended~~.

Configuration Requirements: Base setup 2 for Notification Forwarder object tests with TR lifetime sufficient for this test.

Test Steps:

1. READ SR1 = Subscribed_Recipients
~~2. CHECK (SR1 = {DEST_OBJ_ID, Recipient D1~~
~~DEST_PROCESS_ID, Process Identifier~~
~~FALSE, Issue Confirmed Notifications~~
~~TR1 Time Remaining where (TR-1) <= TR1 <= TR~~
~~}) One list element~~
~~2~~3. WAIT greater of 2 minutes or *the vendor specified time* ~~twice timing resolution~~
~~3~~4. READ SR2 = Subscribed_Recipients
4. VERIFY (SR2, time-remaining < SR1, time-remaining)
~~5. CHECK (SR2 = {DEST_OBJ_ID, Recipient D1~~
~~DEST_PROCESS_ID, Process Identifier~~
~~FALSE, Issue Confirmed Notifications~~
~~TR2 Time Remaining where TR2 < TR1~~
~~}) One list element~~

BTL-26.1 imp3-3: Add IPv6 BDT Host Name Test and Cleanup [BTLWG-1700]

Overview:

No test exists that explicitly tests a B/IPv6 BDT with a host name.
B/IPv6 is missing a test for BBMD_Accept_FD_Registrations is FALSE.

Tests
Made Test 12.3.11.4 explicitly IPv4.
Added a new test for IPv6 BDT Configuration with a host name.

Changes:

Checklist Changes

None

Test Plan Changes

[Modify 9.8.4, delete existing 12.4.4.3.1 reference and add 2 new tests at end of table]

9.8.4 Is Able to Operate in BBMD Mode

The IUT supports BBMD mode.

…		
~~**135.1-2025 - 12.4.4.3.1 - Verify writability of the BDT**~~		
	~~**Test Conditionality**~~	~~Must be executed.~~
	~~**Test Directives**~~	
	~~**Testing Hints**~~	
…		
BTL - 12.4.4.3.X1 - IPv6 - Broadcast Distribution Table Configuration via Host Name Entries		
	Test Conditionality	*Must be executed.*
	Test Directives	
	Testing Hints	
135.1-2025 - 12.3.11.3 - Negative Foreign Device Registration when BBMD_Accept_FD_Registrations is FALSE		
	Test Conditionality	*Must be executed.*
	Test Directives	
	Testing Hints	

[Change 135.1-2025 - 12.3.11.4 to Specified tests and rename test]

10.7.1 Base Requirements

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

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for Data Link Layer - *IPv4*~~(IPv4 or IPv6)~~ and 'Is Able to Operate in BBMD Mode'.
	Testing Hints	
135.1-2025 - 12.3.11.1 - Broadcast-Distribution-Table Holds at Least 5 Entries		
	Test Conditionality	Must be executed.

	Test Directives	
	Testing Hints	
~~**135.1-2025 - 12.3.11.4 - Broadcast Distribution Table Configuration via Hostname Entries**~~ ***BTL - 12.3.11.4 - IPv4 - Broadcast Distribution Table Configuration via Host Name Entries***		
	Test Conditionality	If the IUT claims Protocol_Revision >= 17, this test shall be executed.
	Test Directives	
	Testing Hints	

Specified Test Changes

[Change 135.1-2025 - 12.3.11.4 and put into BTL Specified Tests]

12.3.11.4 *IPv4 -* Broadcast Distribution Table Configuration via ~~Hostname~~ *Host Name* Entries

Reason for Change: *Renamed test, small fixes, and fixed hostname to match standard.*

Purpose: Verify that the IUT accepts and resolves ~~hostname~~ host name entries in the BBMD_Broacast_Distribution_Table and that the resolved IP ~~address~~ *addresses* 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~~ *host name*, HN1 that resolves to an IP address, IP2, and an entry with a non-resolvable ~~hostname~~ *host name* (HN2). Send a broadcast that the IUT should distribute to its peer BBMDs and verify that it sends 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 *D1*~~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~~ *host name*),
   HN2 (an entry with a non-resolvable ~~hostname~~ *host name*))
2. READ BDT = BBMD_Broadcast_Distribution_Table
3. ~~CHECK~~ *VERIFY* (BDT contains IUT, IP1, HN1, HN2 in any order)
4. TRANSMIT ReinitializeDevice-Request
   'Reinitialized State of Device' = ACTIVATE_CHANGES
5. WAIT **Activate Changes Fail Time**
6. WAIT until the IUT completes DNS resolution
7. TRANSMIT
   DA = Local IP Broadcast,
   SA = D1,
   Original-Broadcast-NPDU,
   NPDU = Who-Is-Request
8. RECEIVE
   DA = IP1,
   SA = IUT,
   Forwarded-NPDU,
   Originating-Device = D1,
   NPDU = Who-Is
9. RECEIVE
   DA = IP2,
   SA = IUT,

Forwarded-NPDU,
Originating-Device = D1,
NPDU = Who-Is

10. READ BDT = BBMD_Broadcast_Distribution_Table
11. ~~CHECK~~ *VERIFY* (BDT contains IUT, IP1, HN1, HN2 in any order)
12. TRANSMIT
    DA = IUT,
    SA = D1,
    Read-Broadcast-Distribution-Table
13. 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~~ *host name* entry,
    X'000000000000' for the non-resolvable entry
    -- in the same order as read from the BBMD_Broadcast_Distribution_Table)

[Remove 135.1-2025 - 12.4.4.3.1 - Verify writability of the BDT]

[Add new test to BTL Specified Tests]

12.4.4.3.X1 IPv6 - Broadcast Distribution Table Configuration via Host Name Entries

Reason for Change: No test for this functionality.

Purpose: Verify that the IUT accepts and resolves host name entries in the BBMD_Broadcast_Distribution_Table.

Test Concept: Fill the BBMD_Broadcast_Distribution_Table with 3 entries: an entry with an IPv6 address (IP1), an entry with a resolvable hostname host name, HN1 that resolves to an IPv6 address, IP2, and an entry with a non-resolvable hostname host name (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 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 =(3 entries:
   IP1 (an entry with an IPv6 address),
   HN1 (an entry with a resolvable host name),
   HN2 (an entry with a non-resolvable host name))
2. READ BDT = BBMD_Broadcast_Distribution_Table
3. VERIFY (BDT contains IP1, HN1, HN2 in any order)
4. TRANSMIT ReinitializeDevice-Request
   'Reinitialized State of Device' = ACTIVATE_CHANGES
5. WAIT Activate Changes Fail Time
6. WAIT until the IUT completes DNS resolution
7. TRANSMIT
   DA = Local IP Broadcast,
   SA = D1,
   Original-Broadcast-NPDU,
   NPDU = Who-Is-Request
8. RECEIVE
   DA = IP1,
   SA = IUT,
   Forwarded-NPDU,
   Originating-Device = D1,
   NPDU = Who-Is

9. RECEIVE
   DA = IP2,
   SA = IUT,
   Forwarded-NPDU,
      Originating-Device = D1,
      NPDU = Who-Is
10. READ BDT = BBMD_Broadcast_Distribution_Table
11. VERIFY (BDT contains IP1, HN1, HN2 in any order)

BTL-26.1 imp3-4: Change of State with Commandable Objects [BTLWG-1710]

Overview:

Test 7.3.1.8 does not consider commandable objects.
The Test plan does not verify Local_Time and Local_Date are present.

Clause 12.7.14 Change_Of_State_Time
This property, of type BACnetDateTime, represents the date and time at which the most recent change of state occurred.

Changes:

Checklist Changes

None

Test Plan Changes

[Modify 3.5.4, 3.6.5, 3.7.3]

3.5.4 Supports Change Of State Tracking

The IUT contains or can be made to contain an object with the Change_Of_State_Time, Change_Of_State_Count and Time_Of_State_Count_Reset properties.

~~135.1-2025~~*BTL* - 7.3.1.8 - Change of State Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
135.1-2025 - 7.3.1.24 - Non-zero Writable State Count Test		
	Test Conditionality	If no Binary Input object contains a writable Change_Of_State_Count that accepts writes of non-zero values then this test shall be skipped. If IUT claims Protocol_Revision less than 16, then this test shall be skipped.
	Test Directives	This test shall be performed using a Binary Input object.
	Testing Hints	
Verify EPICS		
	Test Conditionality	*Must be executed.*
	Test Directives	*Verify in the EPICS that the Local_Date and Local_Time properties are present in the Device Object.*
	Testing Hints	

3.6.5 Supports Change Of State Tracking

The IUT contains or can be made to contain and object with the Change_Of_State_Time, Change_Of_State_Count and Time_Of_State_Count_Reset properties.

~~135.1-2025~~*BTL* - 7.3.1.8 - Change of State Test		
	Test Conditionality	Must be executed.
	Test Directives	This test shall be performed using a Binary Output Object.
	Testing Hints	
135.1-2025 - 7.3.1.24 - Non-zero Writable State Count Test		
	Test Conditionality	If no Binary Output object contains a writable Change_Of_State_Count that accepts writes of non-zero values then this test shall be skipped. If IUT claims Protocol_Revision less than 16, then this test shall be skipped.
	Test Directives	This test shall be performed using a Binary Output object.

	Testing Hints	
Verify EPICS		
	Test Conditionality	*Must be executed.*
	Test Directives	*Verify in the EPICS that the Local_Date and Local_Time properties are present in the Device Object.*
	Testing Hints	

3.7.3 Supports Change Of State Tracking

The IUT contains or can be made to contain an object with the Change_Of_State_Time, Change_Of_State_Count and Time_Of_State_Count_Reset properties.

~~***135.1-2025***~~***BTL*** **- 7.3.1.8 - Change of State Test**		
	Test Conditionality	Must be executed.
	Test Directives	This test shall be performed using a Binary Value object.
	Testing Hints	
135.1-2025 - 7.3.1.24 - Non-zero Writable State Count Test		
	Test Conditionality	If no Binary Value object contains a writable Change_Of_State_Count that accepts writes of non-zero values then this test shall be skipped. If IUT claims Protocol_Revision less than 16, then this test shall be skipped.
	Test Directives	This test shall be performed using a Binary Value object.
	Testing Hints	
Verify EPICS		
	Test Conditionality	*Must be executed.*
	Test Directives	*Verify in the EPICS that the Local_Date and Local_Time properties are present in the Device Object.*
	Testing Hints	

Specified Test Changes

[Move test 7.3.1.8 from 135.1-2025 into BTL Specified Tests, and modify]

7.3.1.8 Change of State Tests

Reason For Change: Added Note to Tester about commandable objects.

Purpose: To verify that the properties of objects that collectively track state changes (changes in Present_Value) function as required.

Test Concept: The Present_Value of the object under test *(O1)* is changed. The Change_Of_State_Count property is checked to verify that it has been incremented, and the Change_Of_State_Time property is checked to verify that it has been updated. The Change_Of_State_Count is reset and Time_Of_State_Count_Reset is checked to verify that it has been updated appropriately.

Configuration Requirements: The object being tested shall be configured such that the Present_Value and Change_Of_State_Count properties are writable, or another means of changing these properties shall be provided.

Notes To Tester: If O1 is a commandable object, ensure the Property Array Index field in the write requests is sufficient to cause the Present_Value property to be updated. Consider using a Property Array Index at a priority numerically lower (higher priority) than priority 6 if Minimum_On_Time and Minimum_Off_Time properties are present.

Test Steps:

1. READ PV = Present_Value
2. READ N = Change_of_State_Count

3. IF (PV = ACTIVE) THEN
4. IF (Present_Value is writable) THEN
5. WRITE Present_Value = INACTIVE
6. VERIFY Present_Value = INACTIVE
ELSE
7. MAKE (Present_Value = INACTIVE)
ELSE
8. IF (Present_Value is writable) THEN
9. WRITE Present_Value = ACTIVE
10. VERIFY Present_Value = ACTIVE
ELSE
11. MAKE (Present_Value = ACTIVE)
12. VERIFY (Change_of_State_Count = N+1)
13. VERIFY (Change_Of_State_Time ~= the current local date and time)
14. IF (Change_of_State_Count is writable) THEN
15. WRITE Change_of_State_Count = 0
ELSE
16. MAKE (Change_of_State_Count = 0)
17. VERIFY Time_Of_State_Count_Reset ~= (the current local date and time)

BTL-26.1 imp3-5: BBMD Supporting NAT Must be a Router [BTLWG-1711]

Overview:

A BBMD that supports NAT must also be a router. See Clause J.7.5 (a).

Changes:

Checklist Changes

None

Test Plan Changes

9.3.8 Supports Network Address Translation in BBMD Mode

The IUT is capable of ~~operating behind a router~~ providing Network Address Translation as described in *J.7.5 of the standard*~~addendum 135-2008o-1~~.

135.1-2025 - 12.3.7.1.2 - Broadcast Message from Directly Connected IP Subnet (Two-hop Distribution)		
	Test Conditionality	Must be executed.
	Test Directives	Internet Routers and the IUT shall be configured for NAT.
	Testing Hints	
135.1-2025 - 12.3.7.2.2 - Broadcast Message Forwarded by a Peer BBMD (Two-hop Distribution)		
	Test Conditionality	Must be executed.
	Test Directives	Internet Routers and the IUT shall be configured for NAT.
	Testing Hints	
135.1-2025 - 12.3.7.3.2 - Broadcast Message From a Foreign Device (Two-hop Distribution)		
	Test Conditionality	Must be executed.
	Test Directives	Internet Routers and the IUT shall be configured for NAT.
	Testing Hints	
Verify Checklist		
	Test Conditionality	*Must be executed.*
	Test Directives	*Verify that the IUT claims support for section 10.1 Network Management - Routing.*
	Testing Hints	

Specified Test Changes

None

BTL-26.1 imp3-6: Add T-VMMV-I-B Missing Test [BTLWG-1797]

Overview:

Test 9.21.1.13 is in the Testplan for T-VMT-I-B but not T-VMMV-I-B. This issue adds it to T-VMMV-I-B so that it can be run on Trend Log Multiples.

Changes:

Checklist Changes

None

Test Plan Changes

7.7 Trending - Viewing and Modifying Multiple Values - Internal - B

[Add 9.21.1.13 to section 7.7.2 of BTL Testplan]

7.7.2 Supports all forms of ReadRange

The IUT can accept any of the ReadRange options and respond appropriately.

...		
135.1-2025 - 9.21.1.13 - Reading Items with Negative Count and MOREITEMS		
	Test Conditionality	*Must be executed*
	Test Directives	
	Testing Hints	

Specified Test Changes

None

BTL-26.1 imp3-7: Writing NULL to Non-commandable Properties Test Fix [BTLWG-1818]

Overview:

For reference, the only commandable property is Present_Value. Here is a list of the objects that have Present_Value grouped by commandability (based on 135-2024):

Optionally-commandable Present_Value: Analog Value, Binary Value, Multi-state Value, BitString Value, CharacterString Value, Date Value, Date Pattern Value, DateTime Value, DateTime Pattern Value, Large Analog Value, OctetString Value, Integer Value, Time Value, Time Pattern Value, and Positive Integer Value

Always-commandable Present_Value: Access Door, Analog Output, Binary Lighting Output, Binary Output, Lighting Output, Multi-state Outputs, and Channel

Never-commandable Present_Value: Analog Input, Binary Input, Calendar, Command, Group, Life Safety Point Object, Life Safety Zone, Loop, Multi-state Input, Pulse Converter, Schedule, Load Control, Credential Data Input, Global Group, Alert Enrollment, Timer, Accumulator, Staging, Color, Color Temperature

The issue is that the 9.22.1.6 test directives and conditionality only specify “non-commandable”, which means the objects in the above groups “Optionally-commandable” and “Never-commandable” would be tested (if they are also non-commandable and don’t support NULL). This needs to be fixed so that only the objects in “Optionally-commandable” are considered.

The 135-2024 Interpretation Request attached to Jira for BTLWG-1818 clarifies this: “The standard should have restricted the requirement to quietly accept an attempt to relinquish to only those properties that are optionally commandable for instances that are not commandable”.

Changes:

Checklist Changes

None

Test Plan Changes

4.6 Data Sharing - WriteProperty - B

[Modify 9.22.1.6 under section 4.6 in BTL Test Plan]

4.6.1 Base Requirements

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

…		
BTL - 9.22.1.6 - Writing NULL to Non-commandable Properties		
	Test Conditionality	If the IUT claims Protocol_Revision 20, or prior, this test shall be skipped. If the IUT does not contain any writable non-commandable Present_Value properties *which are optionally commandable object type*, this test shall be skipped.
	Test Directives	Repeat the test for each *optionally commandable* object type *that has an instance that is writable and is non-commandable.* ~~with a writable non-commandable Present_Value supported by the IUT which do not support the value NULL.~~
	Testing Hints	*The only objects that contain optionally commandable Present_Value properties are Value objects (135-2024 clause 19.2.1.1).*

Specified Test Changes

[Modify test 9.22.1.6 in BTL Specified Tests]

9.22.1.6 Writing NULL to Non-commandable Properties
Reason for Change: The standard was changed in PR21 to require that devices not return errors when a Relinquish~~NULL~~ is written to writable, *optionally* non-commandable Present_Value properties.

Purpose: This test case verifies that the IUT returns a Result(+) when an attempt is made to *relinquish the Present_Value property, P1, of an instance of an object, O1, that is writable, non-commandable, and is an optionally commandable object type.* ~~relinquish a *writable* non-commandable *Present_Value* property.~~

Test Concept: Write NULL, *at a priority* ~~*to,* to a writable non-commandable *Present_Value* property,~~ P1 in object O1, and verify the IUT returns a Result(+) and does not modify the property.

Test Configuration: *None*.~~P1 shall be a property for which NULL is not an accepted value.~~

Test Steps:

1. READ X = (O1), P1
2. TRANSMIT WriteProperty-Request,
   'Object Identifier' = O1,
   'Property Identifier' = P1,
   'Property Value' = NULL,
   'Priority' = (any valid value)
3. RECEIVE BACnet-SimpleACK-PDU
4. VERIFY (O1), P1 = X
5. *TRANSMIT WriteProperty-Request,*
   'Object Identifier' = O1,
   'Property Identifier' = P1,
   'Property Value' = NULL,
6. *RECEIVE BACnet-SimpleACK-PDU*
7. *VERIFY (O1), P1 = X*

BTL-26.1 imp3-8: Improve Channel Object's Write_Status Test [BTLWG-859]

Overview:

Improve the Channel object's Write_Status test, by testing that when the List_Of_Object_Property_References are empty that Write_Status remains IDLE.

Changes:

Checklist Changes

None

Test Plan Changes

[Change all references for test **7.3.2.34.4 Write_Status Test** from 135.1 to BTL]

Specified Test Changes

[Move Existing test for Channel objects in 135.1-2025 to BTL Specified Tests.]

7.3.2.34.4 Write_Status Test

Reason For Change: Updated to test that when the List_Of_Object_Property_References property is set to empty, the Write_Status property shall remain set to IDLE.

Purpose: To verify that the Write_Status of the Channel object is IDLE when it's List_Of_Object_Property_References is empty and is IN_PROGRESS while the Channel object's Present_Value is being propagated, FAILED when propagation failed, and SUCCESSFUL when propagation succeeds.

Test Concept: The Channel object's List_Of_Object_Property_Refereces is read and verified to be empty. Then, the Channel object's Write_Status is verified to be IDLE. Next, any valid value is written to the Channel object's Present_Value and Write_Status is verified to be IDLE still. An object property reference, R1, that the IUT cannot reach is set into the List_Of_Object_Reference and then any valid value is written to the Present_Value and the Write_Status is verified to be IN_PROGRESS. After the IUT determines that the referenced device is offline, the Write_Status is verified to be FAILED. ~~Finally, any valid reference~~ *Next, a valid object property reference*, R2, that will cause successful value propagation is set to the List_Of_Object_Property_References and the test is repeated to verify that Write_Status becomes SUCCESSFUL.

Finally, the List_Of_Object_Property_References is set to empty, and the Write_Status is then verified to remain IDLE.

Test Steps:

1. READ LEN = List_Of_Object_Property_References, ARRAY INDEX = 0

-- IDLE test
2. READ L = List_Of_Object_Property_References
3. VERIFY L = (empty)
4. VERIFY Write_Status = IDLE
5. WRITE Present_Value = (any valid value)
6. VERIFY Write_Status = IDLE

-- IN_PROGRESS and FAIL test

7. WRITE List_Of_Object_Property_References = (R1) -- write the whole array
8. WRITE Present_Value = (any valid value)
9. VERIFY Write_Status = IN_PROGRESS
10. WAIT (Channel Write Fail Time * LEN)
11. VERIFY Write_Status = FAILED

-- SUCCESSFUL test
12. WRITE List_Of_Object_Property_References = (R2) -- write the whole array
13. WRITE Present_Value = (any valid value)
14. WAIT (Channel Write Fail Time * LEN)
15. VERIFY Write_Status = SUCCESSFUL

-- IDLE test
16. WRITE List_Of_Object_Property_References = (empty array)
17. VERIFY Write_Status = IDLE

BTL-26.1 imp3-9: Active_COV_Subscriptions SubscribeCOVProperty Test Fixes [BTLWG-1731]

Overview:

7.3.2.10.1 Allows a numeric entry in Active_COV_Subscriptions to not have an increment and disallows a non-numeric from containing an increment.

7.3.2.10.2 Active_COV_Subscriptions SubscribeCOVProperty Test needs to be added to Test Plan 4.20 Base Requirements. This test also references 7.3.2.10.1 when it should be an independent test.

History of 2016bv.
Was renamed to 2020bv with the cov section removed and added to the 2016 erratum document (87).

Erratum 87) Clarify COV Increment in active COV subscriptions

Add note to explain this is from an IC

Changes:

Checklist Changes

None

Test Plan Changes

[Add 135.1-2025 - 7.3.2.10.2 to 4.20.1]

4.20 Data Sharing - Change Of Value Property - B

4.20.1 Base Requirements

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

...		
BTL - 7.3.2.10.2 - Active_COV_Subscriptions SubscribeCOVProperty Test		
	Test Conditionality	*Must be executed.*
	Test Directives	
	Testing Hints	

[Change all occurrences of 7.3.2.10.1 from 135.1 to BTL]
[Change all occurrences of 7.3.2.10.2 from 135.1 to BTL]

Specified Test Changes

7.3.2.10.1 Active_COV_Subscriptions SubscribeCOV Test

Reason for change: Fixed the expected response for numeric and non-numeric.

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: INC_1, INC_2, and INC_3 are each present if the property is numeric and optionally present otherwise.~~

Configuration Requirements: In this test, the tester shall choose three standard objects, O_1, O_2, and O_3, for which the device supports SubscribeCOV. O_1, O_2, and O_3 are not required to refer to different objects. The tester shall also choose three non-zero unique process identifiers, P_1, P_2, and P_3, and three non-zero lifetimes L_1, L_2, and L_3. Lifetime L_1 shall be long enough to allow the initial part of the test to run through to step ~~21~~*23*. Lifetimes L_2 and L_3 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' = P_1,
   'Monitored Object Identifier' = O_1,
   'Issue Confirmed Notifications' = TRUE,
   'Lifetime' = L_1
2. RECEIVE BACnet-SimpleACK-PDU
3. BEFORE **Notification Fail Time**
4. RECEIVE ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = P_1,
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = O_1,
   'Time Remaining' = (a value approximately equal to L_1),
   'List of Values' = (values appropriate to the object type of the monitored object)
5. TRANSMIT BACnet-SimpleACK-PDU
6. IF *O_1, Present_Value*~~P_1~~ is numeric THEN
7. VERIFY Active_COV_Subscriptions = {*S_1*{{TD, P_1}, {O_1, Present_Value}, TRUE, (a value less than L_1), (INC_1 : ~~not present or~~ a valid Increment)}}
   ELSE
8. VERIFY Active_COV_Subscriptions = {*S_1*{{TD, P_1}, {O_1, Present_Value}, TRUE, (a value less than L_1), (INC_1: not present *or any Real value*)}}
9. TRANSMIT SubscribeCOV-Request,
   'Subscriber Process Identifier' = P_2,
   'Monitored Object Identifier' = O_2,
   'Issue Confirmed Notifications' = FALSE,
   'Lifetime' = L_2
10. RECEIVE BACnet-SimpleACK-PDU
11. BEFORE **Notification Fail Time**
12. RECEIVE UnconfirmedCOVNotification-Request,
    'Subscriber Process Identifier' = P_2,
    'Initiating Device Identifier' = IUT,
    'Monitored Object Identifier' = O_2,
    'Time Remaining' = (a value approximately equal to L_2),
    'List of Values' = (values appropriate to the object type of the monitored object)
13. *IF O_2, Present_Value is numeric THEN*
14. VERIFY Active_COV_Subscriptions = {*S_1*~~{ {TD, P_1}, {O_1, Present_Value}, TRUE, (a value less than L_1), INC_1}~~,
    S_2{{TD, P_2}, {O_2, Present_Value}, FALSE, (a value less than L_2), (INC_2: ~~not present if the property is not numeric; present if~~ a valid Increment ~~was provided in the subscription; optionally present otherwise~~)}}
    ELSE
15. *VERIFY Active_COV_Subscriptions = {S_1, S_2{{TD, P_2}, {O_2, Present_Value}, FALSE, (a value less than L_2), (INC_2: not present or any Real value)}}*
16. TRANSMIT SubscribeCOV-Request,
    'Subscriber Process Identifier' = P_3,
    'Monitored Object Identifier' = O_3,
    'Issue Confirmed Notifications' = FALSE,
    'Lifetime' = L_3
17. RECEIVE BACnet-SimpleACK-PDU

18. BEFORE **Notification Fail Time**
19. RECEIVE UnconfirmedCOVNotification-Request,
'Subscriber Process Identifier' = P_3,
'Initiating Device Identifier' = IUT,
'Monitored Object Identifier' = O_3,
'Time Remaining' = (a value approximately equal to L_3),
'List of Values' = (values appropriate to the object type of the monitored object)
20. IF *O_3, Present_Value* ~~P_3~~ is numeric THEN
21. VERIFY Active_COV_Subscriptions = {~~*S_1*{{TD, P_1}, {O_1, Present_Value}, TRUE, (a value less than L_1), INC_1}~~,
S_2~~{{TD, P_2}, {O_2, Present_Value}, FALSE, (a value less than L_2), INC_2}~~,
S_3{{TD, P_3}, {O_3, Present_Value}, FALSE, (a value less than L_3), INC_3: : ~~not present or~~ a valid Increment}}
ELSE
22. VERIFY Active_COV_Subscriptions = {~~*S_1*{{TD, P_1}, {O_1, Present_Value}, TRUE, (a value less than L_1), INC_1}~~,
S_2~~{{TD, P_2}, {O_2, Present_Value}, FALSE, (a value less than L_2), INC_2}~~,
S_3{{TD, P_3}, {O_3, Present_Value}, FALSE, (a value less than L_3), (INC_3: not present *or any Real value*)}}
23. WAIT L_1 + the IUT's timer granularity
24. VERIFY Active_COV_Subscriptions = {*S_2, S_3*~~{{TD, P_2}, {O_2, Present_Value}, FALSE, (a value less than L_2), INC_2 (a valid Increment if the property is REAL)}, {{TD, P_3}, {O_3, Present_Value}, FALSE, (a value less than L_3), INC_3 (a valid Increment if the property is REAL)}~~}
25. TRANSMIT SubscribeCOV-Request,
'Subscriber Process Identifier' = P_3,
'Monitored Object Identifier' = O_3
26. RECEIVE BACnet-SimpleACK-PDU
27. VERIFY Active_COV_Subscriptions = {~~*S_2*{{TD, P_2}, {O_2, Present_Value}, FALSE, (a value less than L_2), INC_2 (a valid Increment if the property is REAL)}~~}
28. TRANSMIT SubscribeCOV-Request,
'Subscriber Process Identifier' = P_2,
'Monitored Object Identifier' = O_2
29. RECEIVE BACnet-SimpleACK-PDU
30. VERIFY Active_COV_Subscriptions = { }

7.3.2.10.2 Active_COV_Subscriptions SubscribeCOVProperty Test

Reason for change: Moved test steps into the test instead of referencing 7.3.2.10.1.

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 SubscribeCOVProperty.

Configuration Requirements: In this test, the tester shall choose three objects and properties, O_1, O_2, and O_3, for which the device supports SubscribeCOVProperty. O_1, O_2, and O_3 are not required to refer to different objects. The tester shall also choose three non-zero unique process identifiers, P_1, P_2, and P_3, and three non-zero lifetimes, L_1, L_2 and L_3. Lifetime L_1 shall be long enough to allow the initial part of the test to run through to step *23*~~14~~. Lifetimes L_2 and L_3 shall be long enough for the whole test to be completed without expiring.

~~Test Steps: The test steps for this test case are identical to the test steps in Clause 7.3.2.10.1 except that SubscribeCOVProperty is used instead of SubscribeCOV and the Active_COV_Subscriptions entries shall reflect the property actually subscribed to (and not Present_Value if the subscribed property is not Present_Value).~~

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

Test Steps:

1. TRANSMIT SubscribeCOVProperty-Request,

'Subscriber Process Identifier' = P_1,
'Monitored Object Identifier' = O_1,
'Issue Confirmed Notifications' = TRUE,
'Lifetime' = L_1
'Monitored Property Identifier' = MP_1,
'COV Increment' = (any valid value -- optional)

2. RECEIVE BACnet-SimpleACK-PDU
3. BEFORE **Notification Fail Time**
4. RECEIVE ConfirmedCOVNotification-Request,
'Subscriber Process Identifier' = P_1,
'Initiating Device Identifier' = IUT,
'Monitored Object Identifier' = O_1,
'Time Remaining' = (a value approximately equal to L_1),
'List of Values' = (value of MP_1 and the value of Status_Flags if present in O_1)
5. TRANSMIT BACnet-SimpleACK-PDU
6. IF P_1 is numeric THEN
7. VERIFY Active_COV_Subscriptions = {S_1{{TD, P_1}, {O_1, MP_1}, TRUE, (a value less than L_1), (INC_1: a valid Increment)}}

ELSE

8. VERIFY Active_COV_Subscriptions = {S_1{{TD, P_1}, {O_1, MP_1}, TRUE, (a value less than L_1), (INC_1: not present or any Real value)}}
9. TRANSMIT SubscribeCOVProperty -Request,
'Subscriber Process Identifier' = P_2,
'Monitored Object Identifier' = O_2,
'Issue Confirmed Notifications' = FALSE,
'Lifetime' = L_2
'Monitored Property Identifier' = MP_2,
'COV Increment' = (any valid value -- optional)
10. RECEIVE BACnet-SimpleACK-PDU
11. BEFORE **Notification Fail Time**
12. RECEIVE UnconfirmedCOVNotification-Request,
'Subscriber Process Identifier' = P_2,
'Initiating Device Identifier' = IUT,
'Monitored Object Identifier' = O_2,
'Time Remaining' = (a value approximately equal to L_2),
'List of Values' = (value of MP_2 and the value of Status_Flags if present in O_2)
13. IF P_2 is numeric THEN
14. VERIFY Active_COV_Subscriptions = {S_1, S_2{{TD, P_2}, {O_2, MP_2}, FALSE, (a value less than L_2), (INC_2: a valid Increment)}}

ELSE

15. VERIFY Active COV Subscriptions = {S_1, S_2{{TD, P_2}, {O_2, MP_2}, FALSE, (a value less than L_2), (INC_2: not present or any Real value)}}
16. TRANSMIT SubscribeCOVProperty-Request,
'Subscriber Process Identifier' = P_3,
'Monitored Object Identifier' = O_3,
'Issue Confirmed Notifications' = FALSE,
'Lifetime' = L_3
'Monitored Property Identifier' = MP_3,
'COV Increment' = (any valid value -- optional)
17. RECEIVE BACnet-SimpleACK-PDU
18. BEFORE **Notification Fail Time**
19. RECEIVE UnconfirmedCOVNotification-Request,
'Subscriber Process Identifier' = P_3,
'Initiating Device Identifier' = IUT,
'Monitored Object Identifier' = O_3,
'Time Remaining' = (a value approximately equal to L_3),
'List of Values' = (value of MP_3 and the value of Status_Flags if present in O_3)
20. IF P_3 is numeric THEN

21. VERIFY Active_COV_Subscriptions = {S_1, S_2, S_3{{TD, P_3}, {O_3, MP_3}, FALSE, (a value less than L_3), INC_3: a valid Increment}}

ELSE

22. VERIFY Active_COV_Subscriptions = {S_1, S_2, S_3{{TD, P_3}, {O_3, MP_3}, FALSE, (a value less than L_3), (INC_3: not present or any Real value)}}

23. WAIT L_1 + the IUT's timer granularity
24. VERIFY Active_COV_Subscriptions = {S_2, S_3}
25. TRANSMIT SubscribeCOVProperty-Request,
    'Subscriber Process Identifier' = P_3,
    'Monitored Object Identifier' = O_3
    'Monitored Property Identifier' = MP_3,
    'COV Increment' = (any valid value -- optional)
26. RECEIVE BACnet-SimpleACK-PDU
27. VERIFY Active_COV_Subscriptions = {S_2}
28. TRANSMIT SubscribeCOVProperty-Request,
    'Subscriber Process Identifier' = P_2,
    'Monitored Object Identifier' = O_2
    'Monitored Property Identifier' = MP_2,
    'COV Increment' = (any valid value -- optional)
29. RECEIVE BACnet-SimpleACK-PDU
30. VERIFY Active_COV_Subscriptions = { }

BTL-26.1 imp3-10: Add Test for Unique Object Names [BTLWG-1829]

Overview:

135-2024, Clause 12.1.1.2 states "…Object_Name properties shall be unique within the BACnet device that maintains them…"

Add existing test 9.22.2.8 to DS-WP-B.

Changes:

Checklist Changes

None

Test Plan Changes

[Add test 135.1-2025 - 9.22.2.8 to Test Plan Clause 4.6.1]

4.6 Data Sharing - WriteProperty - B

4.6.1 Base Requirements

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

...		
135.1-2025 - 9.22.2.8 - Writing An Object_Name With A Value That Is Already In Use		
	Test Conditionality	Must be executed if the IUT contains any objects with a writable Object_Name property.
	Test Directives	
	Testing Hints	

Specified Test Changes

None

BTL-26.1 imp3-11: Remove Physical Check from Tests [BTLWG-419]

Overview:

This proposal removes test steps for checking the physical output. Checks for physical output are not conducive to automation and often require extra configuration yet the behavior of the physical output has no impact on interoperability so the extra cost/effort is not justified.

Changes:

Checklist Changes

Access Door Object		
	R	Base Requirements
	R	Supports command prioritization
	S	Supports configurable Out_Of_Service property
	~~C[1]~~	~~Supports Door_Status property~~
	O	Supports Lock_Status property
	O	Supports Secured_Status property
	O	Supports Door_Unlock_Delay_Time property
	O	Supports Masked_Alarm_Values property
	O	Supports intrinsic reporting
	O	Supports the value source mechanism.
	O[2]*[1]*	Supports configurable Reliability_Evaluation_Inhibit property
	~~[1] If Secured_Status is supported, this is required.~~ [2]*[1]* Protocol_Revision 13 or higher must be claimed. IUT shall not claim this functionality if the only method of generating a fault condition is by writing to the Reliability property.	

Binary Output Object		
	R	Base Requirements
	R	Supports command prioritization
	S	Supports configurable Out_Of_Service property
	~~O~~	~~Supports writable Polarity property~~
	O	Supports change of state tracking
	O	Supports elapsed active time tracking
	O	Supports Minimum_Off_Time
	O	Supports Minimum_On_Time
	O	Supports the value source mechanism.

Binary Lighting Output Object		
	R	Base Requirements
	R	Supports command prioritization
	S	Supports configurable Out_Of_Service property
	O	Supports blink-warn
	~~O~~	~~Supports writable Polarity property~~
	O	Supports strike count tracking
	O	Supports elapsed active time tracking
	O	Supports the value source mechanism.
	O[1,2]	Supports Color_Reference property
	O[1]	Supports color override
	O[3]	Supports configurable Reliability_Evaluation_Inhibit property
	O	Supports intrinsic reporting
	[1] Protocol_Revision 24 or higher must be claimed [2] Must be claimed if the IUT supports color override [3] Protocol_Revision 13 or higher must be claimed. IUT shall not claim this functionality if the only method of generating a fault condition is by writing to the Reliability property.	

Test Plan Changes

~~3.6.4 Supports Writable Polarity Property~~

~~The IUT supports a writable Polarity property in the Binary Output object.~~

~~135.1-2025 - 7.3.2.6.3 - Polarity Property Tests~~		
	~~Test Conditionality~~	~~Must be executed.~~
	~~Test Directives~~	
	~~Testing Hints~~	

3.42.1 Base Requirements

Base requirements must be met by any IUT that supports Access Door objects. *There are no base requirements for this object.*

~~135.1-2025 - 7.3.2.40.1 - Commandable Present_Value Test~~		
	~~Test Conditionality~~	~~Must be executed.~~
	~~Test Directives~~	
	~~Testing Hints~~	

~~3.42.4 Supports Door_Status Property~~

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

~~135.1-2025 - 7.3.2.40.3 - Door_Status with Physical Door Status Tests~~		
	~~Test Conditionality~~	~~If the Door_Status property is permanently configured to have the value UNUSED then this test shall be skipped.~~
	~~Test Directives~~	
	~~Testing Hints~~	

3.42.5 Supports Lock_Status Property

The IUT contains or can be made to contain *an Access Door object with the* Lock_Status property ~~which is writable when Out_Of_Service is True~~.

~~135.1-2025 - 7.3.2.40.4 - Lock_Status Tests~~		
	~~Test Conditionality~~	~~If the physical lock cannot be manipulated without writing to Present_Value of the associated Access Door objet then this test shall be skipped.~~
	~~Test Directives~~	
	~~Testing Hints~~	
BTL - 7.3.2.40.X1 – Present_Value and Lock_Status Test		
	Test Conditionality	*Must be executed.*
	Test Directives	
	Testing Hints	

3.42.7 Supports Door_Unlock_Delay_Time Property

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

~~135.1-2025~~***BTL*** **- 7.3.2.40.6 - Door_Unlock_Delay_Time Test**		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

~~3.55.5 Supports Writable Polarity Property~~

~~The IUT supports a writable Polarity property in the Binary *Lighting* Output object.~~

~~135.1-2025*BTL*~~ - 7.3.2.6.3 - Polarity Property Tests		
	~~Test Conditionality~~	~~Must be executed.~~
	~~Test Directives~~	
	~~Testing Hints~~	

3.9.1 Base Requirements

Base requirements must be met by any IUT that can contain Command objects.

~~135.1-2025*BTL*~~ - 7.3.2.9.2 - Quit on Failure Test		
	Test Conditionality	Must be executed
	Test Directives	
	Testing Hints	
~~135.1-2025*BTL*~~ - 7.3.2.9.4 - Empty Action List Test		
	Test Conditionality	Must be executed
	Test Directives	
	Testing Hints	
~~135.1-2025*BTL*~~ - 7.3.2.9.5 - Action 0 Test		
	Test Conditionality	Must be executed
	Test Directives	
	Testing Hints	
~~135.1-2025*BTL*~~ - 7.3.2.9.7 - Write While In_Process is TRUE Test.		
	Test Conditionality	Must be executed
	Test Directives	
	Testing Hints	

3.9.3 Supports Post_Delay

The properties of binary objects that collectively track state changes function as required.

~~135.1-2025*BTL*~~ - 7.3.2.9.1 - All Writes Successful with Post Delay Test		
	Test Conditionality	If the prescribed test can be configured, it must be executed.
	Test Directives	
	Testing Hints	

3.54.5 Supports Blink-Warn

The Blink_Warn_Enable property in Lighting Output is writable or can be changed to TRUE by other means.

~~135.1-2025*BTL*~~ - 7.3.1.27.1 - Blink-Warn WARN Command Test		
	Test Conditionality	Must be executed.
	Test Directives	Must be executed using both the Present_Value and Lighting_Command properties.
	Testing Hints	
~~135.1-2025*BTL*~~ - 7.3.1.27.2 - Blink-Warn WARN_OFF Command Test		
	Test Conditionality	Must be executed.
	Test Directives	Must be executed using both the Present_Value and Lighting_Command properties.
	Testing Hints	
~~135.1-2025*BTL*~~ - 7.3.1.27.3 - Blink-Warn WARN_RELINQUISH Command Test		
	Test Conditionality	Must be executed.
	Test Directives	Must be executed using both the Present_Value and Lighting_Command properties.

	Testing Hints	
~~135.1-2025~~*BTL* - 7.3.1.27.4 - Blink-Warn STOP Command Test		
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test with WARN_OFF and WARN_RELINQUISH commands
	Testing Hints	
~~135.1-2025~~*BTL* - 7.3.1.27.5 - Blink-Warn WARN Command Failure Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
~~135.1-2025~~*BTL* - 7.3.1.27.6 - Blink-Warn WARN_OFF Command Failure Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
~~135.1-2025~~*BTL* - 7.3.1.27.7 - Blink-Warn WARN_RELINQUISH Command Failure Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
~~135.1-2025~~*BTL* - 7.3.1.27.8 - Blink-Warn WARN_OFF Command Halted Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
~~135.1-2025~~*BTL* - 7.3.1.27.9 - Blink-Warn WARN_RELINQUISH Command Halted Test		
	Test Conditionality	Must be executed/
	Test Directives	
	Testing Hints	

3.55.4 Supports Blink-Warn

The IUT supports blink-warn in the Binary Lighting Output object. The Blink_Warn_Enable property is true or can be set to true.

~~135.1-2025~~*BTL* - 7.3.1.27.1 - Blink-Warn WARN Command Test		
	Test Conditionality	Must be executed.
	Test Directives	Must be executed using both the Present_Value and Lighting_Command properties.
	Testing Hints	
~~135.1-2025~~*BTL* - 7.3.1.27.2 - Blink-Warn WARN_OFF Command Test		
	Test Conditionality	Must be executed.
	Test Directives	Must be executed using both the Present_Value and Lighting_Command properties.
	Testing Hints	
~~135.1-2025~~*BTL* - 7.3.1.27.3 - Blink-Warn WARN_RELINQUISH Command Test		
	Test Conditionality	Must be executed.
	Test Directives	Must be executed using both the Present_Value and Lighting_Command properties.
	Testing Hints	
~~135.1-2025~~*BTL* - 7.3.1.27.4 - Blink-Warn STOP Command Test		
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test with WARN_OFF and WARN_RELINQUISH commands
	Testing Hints	
~~135.1-2025~~*BTL* - 7.3.1.27.5 - Blink-Warn WARN Command Failure Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
~~135.1-2025~~*BTL* - 7.3.1.27.6 - Blink-Warn WARN_OFF Command Failure Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

~~135.1-2025~~*BTL* - 7.3.1.27.7 - Blink-Warn WARN_RELINQUISH Command Failure Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
~~135.1-2025~~*BTL* - 7.3.1.27.8 - Blink-Warn WARN_OFF Command Halted Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
~~135.1-2025~~*BTL* - 7.3.1.27.9 - Blink-Warn WARN_RELINQUISH Command Halted Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

3.54.6 Supports Transition Property

The IUT contains Lighting Output Objects in which the Transition property is supported.

...		
~~135.1-2025~~*BTL* - 7.3.2.39.11 - Transition Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

3.54.8 Supports Min_Actual_Value and Max_Actual_Value Properties

The IUT contains Lighting Output Objects in which the the Min_Actual_Value and Max_Actual_Value properties are supported.

...		
~~135.1-2025 - 7.3.2.39.14 - Min_Actual_Value and Max_Actual_Value Scaling Test~~		
	~~**Test Conditionality**~~	~~Must be executed.~~
	~~**Test Directives**~~	
	~~**Testing Hints**~~	

Specified Test Changes

[Move these tests from 135.1 and modify as shown]

7.3.1.27.1 Blink-Warn WARN Command Test

Reason for Change: Remove physical blink warn check.

Purpose: To verify the correct operation of the blink-warn WARN command.

Test Concept: Select an object O1 that supports blink-warn WARN command. Ensure O1 is not in egress mode and the specific properties have been configured to support blink-warn. Execute blink-warn WARN command by writing C1 to PROP_REF at a priority PTY1 of O1 and validate the specified blink-warn command functions correctly. Validate the Priority_Array value at priority PTY1 remains.

Configuration Requirements: O1 shall be configured such that all slots in the Priority_Array numerically less than PTY1 have a value of NULL and no internal algorithms are issuing commands to O1 at a priority numerically less than or equal to PTY1. The Priority_Array at PTY1 has a value V1, Blink_Warn_Enable is TRUE, Egress_Active is FALSE.

	Binary Lighting Output object	**Lighting Output object**
PROP_REF	Present_Value	Present_Value or Lighting_Command
C1	WARN	-1.0 if PROP_REF = Present_Value, otherwise WARN
V1	ON	>1.0

Test Steps:

1. VERIFY Priority_ Array = V1, ARRAY INDEX = PTY1
2. VERIFY Blink_Warn_Enable = TRUE
3. VERIFY Egress_Active = FALSE
4. WRITE PROP_REF = C1, PRIORITY = PTY1
5. ~~BEFORE **Internal Processing Fail Time**~~
   ~~CHECK (blink-warn occurred)~~
   -- blink warn occurs
5. VERIFY Egress_Active = FALSE
7. VERIFY Priority_ Array = V1, ARRAY INDEX = PTY1

7.3.1.27.2 Blink-Warn WARN_OFF Command Test

Reason for Change: Remove physical blink warn check.

Purpose: To verify the correct operation of the blink-warn WARN_OFF command.

Test Concept: Select an object O1 that supports blink-warn commands. Ensure O1 is not in egress mode and the specific properties have been configured to support blink-warn. Execute blink-warn WARN_OFF command by writing C1 to PROP_REF at a priority PTY1 of O1 and validate the specified blink-warn command functions correctly. Validate the Priority_Array value at priority PTY1 after Egress_Time seconds has elapsed.

Configuration Requirements: O1 shall be configured such that all slots in the Priority_Array numerically less than PTY1 have a value of NULL and no internal algorithms are issuing commands to O1 at a priority numerically less than or equal to PTY1. The Priority_Array at PTY1 has a value V1, Blink_Warn_Enable is TRUE, Egress_Active is FALSE, and Egress_Time is a non-zero value.

	Binary Lighting Output object	**Lighting Output object**
PROP_REF	Present_Value	Present_Value or Lighting_Command
C1	WARN_OFF	-3.0 if PROP_REF = Present_Value, otherwise WARN_OFF
V1	ON	>1.0
V2	OFF	0.0

Test Steps:

1. VERIFY Priority_ Array = V1, ARRAY INDEX = PTY1
2. VERIFY Blink_Warn_Enable = TRUE
3. VERIFY Egress_Time > 0
4. VERIFY Egress_Active = FALSE
5. WRITE PROP_REF = C1, PRIORITY = PTY1
6. *READ T1.date = Device, IUT, Local_Date*
7. *READ T1.time = Device, IUT, Local_Time*
6. ~~T1 = current local time~~
8. VERIFY Priority_Array = V1, ARRAY INDEX = PTY1
9. *WHILE (Egress_Active = TRUE) {}*
8. ~~WHILE (Egress_Active = TRUE)~~
   ~~WAIT 1s~~
10. *READ T2.date = Device, IUT, Local_Date*
11. *READ T2.time = Device, IUT, Local_Time*
9. ~~T2 = current local time~~
10. ~~CHECK (blink-warn occurred)~~
    -- blink warn occurred
12. VERIFY Egress_Time ~= (T2 – T1)
13. VERIFY Priority_ Array = V2, ARRAY INDEX = PTY1

7.3.1.27.3 Blink-Warn WARN_RELINQUISH Command Test

Reason for Change: Remove physical blink warn check.

Purpose: To verify the correct operation of the blink-warn WARN_RELINQUISH commands.

Test Concept: Select an object O1 that supports blink-warn commands. Ensure O1 is not in egress mode and the specific properties have been configured to support blink-warn. Execute blink-warn WARN_RELINQUISH command by writing C1 to PROP_REF at a priority PTY1 of O1 and validate the specified blink-warn command functions correctly. Validate the Priority_Array value at priority PTY1 after Egress_Time seconds has elapsed.

Configuration Requirements: O1 shall be configured such that all slots in the Priority_Array numerically less than PTY1 shall have a value of NULL, at least one slot numerically greater than PTY1 or Relinquish_Default shall have a value of V2, and all other slots numerically greater than PTY1 shall have a value of V0. No internal algorithms are issuing commands to O1 at any priority. The Priority_Array at PTY1 has a value V1, Blink_Warn_Enable is TRUE, Egress_Time is a non-zero value, and Egress_Active is FALSE.

	Binary Lighting Output object	**Lighting Output object**
PROP_REF	Present_Value	Present_Value or Lighting_Command
C1	WARN_RELINQUISH	-2.0 if PROP_REF = Present_Value, otherwise WARN_OFF
V0	NULL or OFF	NULL or 0.0
V1	ON	>1.0
V2	OFF	0.0

Test Steps:

1. VERIFY Priority_ Array = V1, ARRAY INDEX = PTY1
2. VERIFY Blink_Warn_Enable = TRUE
3. VERIFY Egress_Time > 0
4. VERIFY Egress_Active = FALSE
5. WRITE PROP_REF = C1, PRIORITY = PTY1
6. *READ T1.date = Device, IUT, Local_Date*
7. *READ T1.time = Device, IUT, Local_Time*
6. ~~T1 = current local time~~
7. VERIFY Priority_Array = V1, ARRAY INDEX = PTY1
9. *WHILE (Egress_Active = TRUE) {}*
   -- blink warn occured
8. ~~WHILE (Egress_Active = TRUE)~~
   ~~WAIT 1s~~
9. ~~T2 = current local time~~
10. ~~CHECK (blink-warn occurred)~~
10. *READ T2.date = Device, IUT, Local_Date*
11. *READ T2.time = Device, IUT, Local_Time*
12. VERIFY Egress_Time ~= (T2 – T1)
13. VERIFY Priority_ Array = NULL, ARRAY INDEX = PTY1

7.3.1.27.4 Blink-Warn STOP Command Test

Reason for Change: Remove physical blink warn check.

Purpose: To verify the correct operation of the blink-warn STOP command.

Test Concept: Select an object O1 that supports blink-warn commands. Ensure O1 is not in egress mode and the specific properties have been configured to support blink-warn. Execute blink-warn command by writing C1 to PROP_REF at a priority PTY1 of O1 *causing a blink-warn to occur* ~~and validate that blink-warn occurs~~. Before the Egress_Time times out, STOP the egress process and validate the Priority_Array value at PTY1 remains equal to V1 after Egress_Time.

Configuration Requirements: O1 shall be configured such that all slots in the Priority_Array numerically less than PTY1 shall have a value of NULL, at least one slot numerically greater than PTY1 or Relinquish_Default shall have a value of V2, and all other slots numerically greater than PTY1 shall have a value of V0. No internal algorithms are issuing commands to O1 at a

priority numerically less than or equal to PTY1. The Priority_Array at PTY1 has a value V1, Blink_Warn_Enable is TRUE, Egress_Time is a non-zero value, and Egress_Active is FALSE.

	Binary Lighting Output object	**Lighting Output object**
PROP_REF	Present_Value	Lighting_Command
C1	WARN_RELINQUISH or WARN_OFF	WARN_RELINQUISH or WARN_OFF
V0	NULL or OFF	NULL or 0.0
V1	ON	>1.0
V2	OFF	0.0

Test Steps:

1. READ V1 = Priority Array, ARRAY INDEX = PTY1
~~1. VERIFY Priority_Array = V1, ARRAY INDEX = PTY1~~
2. VERIFY Blink_Warn_Enable = TRUE
3. VERIFY Egress_Time > 0
4. VERIFY Egress_Active = FALSE
5. WRITE PROP_REF = C1, PRIORITY = PTY1
~~6. T1 = current local time~~
~~7. BEFORE **Internal Processing Fail Time***
--(blink-warn occurs)
~~CHECK (blink-warn occurred)~~
6. VERIFY Egress_Active = TRUE
7. WRITE PROP_REF = STOP, PRIORITY = PTY1
~~9. WAIT less than Egress_Time~~
~~WRITE PROP_REF = STOP, PRIORITY = PTY1~~
~~10. T2 = current local time~~
~~8. WAIT **Internal Processing Fail Time***
8. VERIFY Egress_Active = FALSE
~~13. WAIT Egress_Time – (T2 – T1) + **Internal Processing Fail Time***
9. VERIFY Priority_ Array = V1, ARRAY INDEX = PTY1

7.3.1.27.5 Blink-Warn WARN Command Failure Test

Reason for Change: Remove physical blink warn check.

Purpose: To verify blink-warn WARN command does not occur when, the specified priority is not the highest active priority, the value at the specified priority is off or Blink_Warn_Enable is FALSE.

Test Concept: Select an object O1 that supports blink-warn commands. Configure O1 such that a blink-warn command would generate a blink-warn except set the specified failure conditions. Verify blink-warn does not occur and the Priority_Array is not affected.

Configuration Requirements: O1 shall be configured such that all slots in the Priority_Array numerically less than PTY1 have a value of NULL and no internal algorithms are issuing commands to O1 at a priority numerically less than or equal to PTY1. Select a priority, PTY2, which is numerically less than PTY1 and not equal to 6. Blink_Warn_Enable is TRUE, Egress_Active is FALSE.

	Binary Lighting Output object	**Lighting Output object**
PROP_REF	Present_Value	Present_Value or Lighting_Command
C1	WARN	-1.0 if PROP_REF = Present_Value, otherwise WARN
V1, V2	ON	>1.0
V3	OFF	0.0

Test Steps:

-- Test for the specified priority is not the highest active priority

1. VERIFY Blink_Warn_Enable = TRUE
2. WRITE Present_Value = V1, PRIORITY = PTY1
3. VERIFY Egress_Active = FALSE
4. WRITE Present_Value = V2, PRIORITY = PTY2
5. WRITE PROP_REF = C1, PRIORITY = PTY1
6. ~~WAIT **Internal Processing Fail Time**~~
   ~~CHECK (blink-warn-did-not-occur)~~
   -- blink warn does not occur
6. VERIFY Egress_Active = FALSE
7. VERIFY Priority_Array = V1, ARRAY INDEX = PTY1
8. WRITE Present_Value = NULL, PRIORITY = PTY2

-- Test for the value at the specified priority is either OFF or 0.0
9. WRITE Present_Value = V3, PRIORITY = PTY1
10. WRITE PROP_REF = C1, PRIORITY = PTY1
12. ~~WAIT **Internal Processing Fail Time**~~
    ~~CHECK (blink-warn-did-not-occur)~~
    -- blink warn does not occur
11. VERIFY Egress_Active = FALSE
12. VERIFY Priority_ Array = V3, ARRAY INDEX = PTY1
13. WRITE Present_Value = V1, PRIORITY = PTY1

-- Test for Blink_Warn_Enable is FALSE
14. IF (Blink_Warn_Enable is writable) THEN
15. WRITE Blink_Warn_Enable = FALSE
16. WRITE PROP_REF = C1, PRIORITY = PTY1
19. ~~WAIT **Internal Processing Fail Time**~~
    ~~CHECK (blink-warn-did-not-occur)~~
    -- blink warn does not occur
17. VERIFY Egress_Active = FALSE
18. VERIFY Priority_ Array = V1, ARRAY INDEX = PTY1

7.3.1.27.6 Blink-Warn WARN_OFF Command Failure Test

Reason for Change: Remove physical blink warn check.

Purpose: To verify blink-warn WARN_OFF command does not occur when the specified priority is not the highest active priority, the Present_Value is either 0.0 or OFF, or Blink_Warn_Enable is FALSE.

Test Concept: Select an object O1 that supports blink-warn commands. Configure O1 such that a blink-warn command would generate a blink-warn except set the specified failure conditions. Verify blink-warn does not occur and the Priority_Array is correctly changed.

Configuration Requirements: O1 shall be configured such that all slots in the Priority_Array numerically less than PTY1 have a value of NULL and no internal algorithms are issuing commands to O1 at a priority numerically less than or equal to PTY1. Blink_Warn_Enable is TRUE, Egress_Time is a non-zero value and Egress_Active is FALSE.

	Binary Lighting Output object	**Lighting Output object**
PROP_REF	Present_Value	Present_Value or Lighting_Command
C1	WARN_OFF	-3.0 if PROP_REF = Present_Value, otherwise WARN_OFF
V1, V2	ON	>1.0
V3	OFF	0.0

Test Steps:

-- Test for the specified priority is not the highest active priority
1. VERIFY Blink_Warn_Enable = TRUE
2. VERIFY Egress_Time > 0
3. WRITE Present_Value = V1, PRIORITY = PTY1

4. VERIFY Egress_Active = FALSE
5. WRITE Present_Value = V2, PRIORITY = PTY2, a value not equal to 6 and less than PTY1
6. WRITE PROP_REF = C1, PRIORITY = PTY1
7. ~~WAIT **Internal Processing Fail Time**~~
   ~~CHECK (blink-warn did not occur)~~
   -- blink warn does not occur
7. VERIFY Egress_Active = FALSE
8. VERIFY Priority_ Array = V3, ARRAY INDEX = PTY1
9. WRITE Present_Value = V1, PRIORITY = PTY1

-- Test for the Present_Value is OFF or 0.0
10. WRITE Present_Value = V3, PRIORITY = PTY2, a value not equal to 6 and less than PTY1
11. WRITE PROP_REF = C1, PRIORITY = PTY1
13. ~~WAIT **Internal Processing Fail Time**~~
    ~~CHECK (blink-warn did not occur)~~
    -- blink warn does not occur
12. VERIFY Egress_Active = FALSE
13. VERIFY Priority_ Array = V3, ARRAY INDEX = PTY1
14. WRITE Present_Value = NULL, PRIORITY = PTY2
15. WRITE Present_Value = V1, PRIORITY = PTY1

-- Test for Blink_Warn_Enable is FALSE
16. IF (Blink_Warn_Enable is writable) THEN
17. WRITE Blink_Warn_Enable = FALSE
18. WRITE PROP_REF = C1, PRIORITY = PTY1
21. ~~WAIT **Internal Processing Fail Time**~~
    ~~CHECK (blink-warn did not occur)~~
    -- blink warn does not occur
19. VERIFY Egress_Active = FALSE
20. VERIFY Priority_ Array = V3, ARRAY INDEX = PTY1

7.3.1.27.7 Blink-Warn WARN_RELINQUISH Command Failure Test

Reason for Change: Remove physical blink warn check.

Purpose: To verify blink-warn WARN_RELINQUISH command does not occur when the specified priority is not the highest active priority, the value at the specified priority is V0, the value of the next highest non-NULL priority, including Relinquish_Default, is V1, or Blink_Warn_Enable is FALSE.

Test Concept: Select an object O1 that supports blink-warn commands. Configure O1 such that a blink-warn command would generate a blink-warn except set the specified failure conditions. Verify blink-warn does not occur and the Priority_Array is correctly changed.

Configuration Requirements: O1 shall be configured such that all slots in the Priority_Array numerically less than PTY1 shall have a value of NULL, at least one slot numerically greater than PTY1 or Relinquish_Default shall have a value of V2, and all other slots numerically greater than PTY1 shall have a value of V0. No internal algorithms are issuing commands to O1 at any priority. Blink_Warn_Enable is TRUE, Egress_Time is a non-zero value, Egress_Active is FALSE.

	Binary Lighting Output object	**Lighting Output object**
PROP_REF	Present_Value	Present_Value or Lighting_Command
C1	WARN_RELINQUISH	-2.0 if PROP_REF = Present_Value, otherwise WARN_RELINQUISH
V0	OFF or NULL	0.0 or NULL
V1	ON	>1.0
V2	OFF	0.0

Test Steps:

-- Test for the specified priority is not the highest active priority

1. VERIFY Blink_Warn_Enable = TRUE
2. VERIFY Egress_Time > 0
3. WRITE Present_Value = V1, PRIORITY = PTY1
4. VERIFY Egress_Active = FALSE
5. WRITE Present_Value = V1, PRIORITY = PTY2, a value not equal to 6 and less than PTY1
6. WRITE PROP_REF = C1, PRIORITY = PTY1
7. ~~WAIT **Internal Processing Fail Time**~~
   ~~CHECK (blink-warn did not occur)~~
   -- blink warn does not occur
7. VERIFY Egress_Active = FALSE
8. VERIFY Priority_ Array = NULL, ARRAY INDEX = PTY1
9. WRITE Present_Value = NULL, PRIORITY = PTY2

-- Test for the value at the specified priority is OFF or 0.0
10. WRITE Present_Value = V2 PRIORITY = PTY1
11. WRITE PROP_REF = C1, PRIORITY = PTY1
13. ~~WAIT **Internal Processing Fail Time**~~
    ~~CHECK (blink-warn did not occur)~~
    -- blink warn does not occur
12. VERIFY Egress_Active = FALSE
13. VERIFY Priority_ Array = NULL, ARRAY INDEX = PTY1

-- Test for the value at the specified priority is NULL
14. WRITE Present_Value = NULL, PRIORITY = PTY1
15. WRITE PROP_REF = C1, PRIORITY = PTY1
18. ~~WAIT **Internal Processing Fail Time**~~
    ~~CHECK (blink-warn did not occur)~~
    -- blink warn does not occur
16. VERIFY Egress_Active = FALSE
17. VERIFY Priority_ Array = NULL, ARRAY INDEX = PTY1

-- Test for the value of the next highest non-NULL priority is ON or > 1.0
18. WRITE Present_Value = V1 PRIORITY = PTY1
19. WRITE Present_Value = V1, PRIORITY = PTY3, a value numerically greater than PTY1
20. WRITE PROP_REF = C1, PRIORITY = PTY1
24. ~~WAIT **Internal Processing Fail Time**~~
    ~~CHECK (blink-warn did not occur)~~
    -- blink warn does not occur
21. VERIFY Egress_Active = FALSE
22. VERIFY Priority_ Array = NULL, ARRAY INDEX = PTY1
23. WRITE Present_Value = NULL, PRIORITY = PTY3

-- Test for the value of Relinquish_Default is ON or > 1.0
24. IF (Relinquish_Default is writable) THEN
25. WRITE Present_Value = V1, PRIORITY = PTY1
26. WRITE Relinquish_Default = V1
27. WRITE PROP_REF = C1, PRIORITY = PTY1
32. ~~WAIT **Internal Processing Fail Time**~~
    ~~CHECK (blink-warn did not occur)~~
    -- blink warn does not occur
28. VERIFY Egress_Active = FALSE
29. VERIFY Priority_ Array = NULL, ARRAY INDEX = PTY1
30. WRITE Relinquish_Default = V2

-- Test for Blink_Warn_Enable is FALSE
31. IF (Blink_Warn_Enable is writable) THEN
32. WRITE Present_Value = V1, PRIORITY = PTY1
33. WRITE Blink_Warn_Enable = FALSE
34. WRITE PROP_REF = C1, PRIORITY = PTY1
40. ~~WAIT **Internal Processing Fail Time**~~

~~CHECK (blink-warn did not occur)~~
-- blink warn does not occur
35. VERIFY Egress_Active = FALSE
36. VERIFY Priority_ Array = NULL, ARRAY INDEX = PTY1

7.3.1.27.8 Blink-Warn WARN_OFF Command Halted Test

Reason for Change: Remove physical blink warn check.

Purpose: To verify blink-warn WARN_OFF execution is halted when a higher priority entry is written or the Present_Value at the specified priority is changed.

Test Concept: Select an object O1 that supports blink-warn commands. Configure O1 such that a blink-warn command will generate a blink-warn. Before the Egress timer expires, verify the specified actions clear the blink-warn properties and the Priority_Array is correctly changed.

Configuration Requirements: O1 shall be configured such that all slots in the Priority_Array numerically less than PTY1 have a value of NULL and no internal algorithms are issuing commands to O1 at a priority numerically less than or equal to PTY1. Blink_Warn_Enable is TRUE, Egress_Time is a non-zero value and Egress_Active is FALSE.

	Binary Lighting Output object	**Lighting Output object**
PROP_REF	Present_Value	Present_Value or Lighting_Command
C1	WARN_OFF	-3.0 if PROP_REF = Present_Value, otherwise WARN_OFF
V1 to V3	ON	>1.0
V4	OFF	0.0

Test Steps:

-- Test for a higher priority entry is written to a non NULL value
1. WRITE Present_Value = V1, PRIORITY = PTY1
2. VERIFY Blink_Warn_Enable = TRUE
3. VERIFY Egress_Time > 0
4. VERIFY Egress_Active = FALSE
5. WRITE PROP_REF = C1, PRIORITY = PTY1
~~6. BEFORE **Internal Processing Fail Time**~~
~~CHECK (blink-warn occurred)~~
-- blink warn occurs
6. VERIFY Egress_Active = TRUE
7. WRITE Present_Value = V2, PRIORITY = PTY2, a value not equal to 6 and less than PTY1
~~7. BEFORE Egress_Active = FALSE~~
~~WRITE Present_Value = V2, PRIORITY = PTY2, a value not equal to 6 and less than PTY1~~
8. VERIFY Egress_Active = FALSE
9. VERIFY Priority_ Array = V4, ARRAY INDEX = PTY1
10. WRITE Present_Value = NULL, PRIORITY = PTY2

-- Test for the Present_Value at the specified property is changed
11. WRITE Present_Value = V1, PRIORITY = PTY1
~~12. VERIFY Blink_Warn_Enable = TRUE~~
~~13. VERIFY Egress_Time > 0~~
~~14. VERIFY Egress_Active = FALSE~~
12 WRITE PROP_REF = C1, PRIORITY = PTY1
~~15. BEFORE **Internal Processing Fail Time**~~
~~CHECK (blink-warn occurred)~~
-- blink warn occurs
13. VERIFY Egress_Active = TRUE
~~17. BEFORE Egress_Active = FALSE~~
~~WRITE Present_Value = V3, PRIORITY = PTY1~~
14. WRITE Present_Value = V3, PRIORITY = PTY1

15. VERIFY Egress_Active = FALSE
16. VERIFY Priority_ Array = V3, ARRAY INDEX = PTY1

7.3.1.27.9 Blink-Warn WARN_RELINQUISH Command Halted Test

Reason for Change: Remove physical blink warn check.

Purpose: To verify blink-warn WARN_RELINQUISH execution is halted when a higher priority entry is written or the Present_Value at the specified priority is changed.

Test Concept: Select an object O1 that supports blink-warn commands. Configure O1 such that a blink-warn command will generate a blink-warn. Before the Egress timer expires, verify the specified actions clear the blink-warn properties and the Priority_Array is correctly changed.

Configuration Requirements: O1 shall be configured such that all slots in the Priority_Array numerically less than PTY1 have a value of NULL and at least one slot numerically greater than PTY1 or Relinquish_Default shall have a value of V1 and all other slots numerically greater than PTY1 shall have a value of V0. No internal algorithms are issuing commands to O1 at any priority. Blink_Warn_Enable is TRUE, Egress_Time is a non-zero value, Egress_Active is FALSE.

	Binary Lighting Output object	**Lighting Output object**
PROP_REF	Present_Value	Present_Value or Lighting_Command
C1	WARN_RELINQUISH	-2.0 if PROP_REF = Present_Value, otherwise WARN_RELINQUISH
V0	OFF or NULL	0.0 or NULL
V1	OFF	0.0
V2	ON	>1.0
V3	OFF	any value >1.0 and not equal to V2

Test Steps:

-- Test for a higher priority entry is written to a non NULL value
1. WRITE Present_Value = V2, PRIORITY = PTY1
2. VERIFY Blink_Warn_Enable = TRUE
3. VERIFY Egress_Time > 0
4. VERIFY Egress_Active = FALSE
5. WRITE PROP_REF = C1, PRIORITY = PTY1
~~6. BEFORE **Internal Processing Fail Time**~~
~~CHECK (blink-warn occurred)~~
~~-- blink warn occurs~~
6. VERIFY Egress_Active = TRUE
~~7. BEFORE Egress_Active = FALSE~~
~~WRITE Present_Value = V2, PRIORITY = PTY2, a value not equal to 6 and less than PTY1~~
7. WRITE Present_Value = V2, PRIORITY = PTY2, a value not equal to 6 and less than PTY1
8. VERIFY Egress_Active = FALSE
9. VERIFY Priority_ Array = NULL, ARRAY INDEX = PTY1
10. WRITE Present_Value = NULL, PRIORITY = PTY2

-- Test for the Present_Value at the specified property is changed
11. WRITE Present_Value = V2, PRIORITY = PTY1
~~12. VERIFY Blink_Warn_Enable = TRUE~~
~~13. VERIFY Egress_Time > 0~~
~~14. VERIFY Egress_Active = FALSE~~
12. WRITE PROP_REF = C1, PRIORITY = PTY1
~~13. BEFORE **Internal Processing Fail Time**~~
~~CHECK (blink-warn occurred)~~
~~-- blink warn occurs~~
13. VERIFY Egress_Active = TRUE
~~17. BEFORE Egress_Active = FALSE~~
~~WRITE Present_Value = V3, PRIORITY = PTY1~~

14. WRITE Present_Value = V3, PRIORITY = PTY1
15. VERIFY Egress_Active = FALSE
~~19. VERIFY Priority_Array = V3, ARRAY INDEX = PTY1~~

7.3.2.9.1 All Writes Successful with Post Delay Test

Reason for Change: Remove reference to externally visible output.

Purpose: To verify that a Command object can successfully execute an action list that includes post delays.

Test Concept: The IUT is configured with an action list that includes manipulating a sequence of *property values* ~~externally visible outputs~~ with a time delay between each *one* ~~output~~. The TD triggers *the* ~~this~~ action list and *verifies post delays are applied* ~~the tester observes the external changes~~. ~~If the IUT does not support Post Delay, then this test shall be omitted. If the IUT does not support action list configuration for this Test Concept, then this test shall be omitted.~~

Configuration Requirements: The IUT shall be configured with a Command object *O1* having *Action property containing* an action list, X, that includes writing *to property P1, with a value of V1, Post_Delay PD1, and another write to property P1, with a value of V2, and Post Delay PD2. PD1 may equal PD2 but V1 must not equal V2. At the start of the test, P1 has a value V0 that is different than V1.* ~~a sequence of externally visible outputs. There shall be a post delay between writes to the externally visible outputs that is long enough for the tester to observe the delay.~~ *PD1 and PD2 must be configured with a value large enough to perform the VERIFY steps between actions. If the IUT can not meet these configuration requirements, this test shall be skipped.*

Test Steps:
1. WRITE Present_Value = *the array index for* X
~~2. RECEIVE BACnet-SimpleACK-PDU~~
~~4. CHECK (for the externally visible actions and verify that there is a post delay)~~
-- (for any externally visible actions, verify that there is a post delay)
2. BEFORE PD1 {
3. VERIFY In_Process = TRUE
4. *VERIFY P1 = V1*
}
5. WAIT (the remainder of PD1)
6. BEFORE PD2
7. VERIFY In_Process = TRUE
8. VERIFY P1 = V2
9. WAIT (the remainder of PD2)
10. VERIFY In_Process = FALSE
11. VERIFY All_Writes_Successful = TRUE

7.3.2.9.2 Quit on Failure Test

Reason for Change: Remove reference to externally visible output.

Purpose: To verify that a Command object can successfully execute Quit_On_Failure procedures.

Test Concept: *The TD writes to the Present_Value of a command object O1 in the IUT that triggers the execution of an action list with multiple write operations, the first of which will fail and Quit On Failue is TRUE. The TD makes another write to Present_Value, triggering execution of another action list with multiple write operations, the first of which will fail and Quit_On_Failure is FALSE. The actions that occur after the failed write operation are observed to verify that they do not occur when Quit On Failure is TRUE and occur when Quit On Failure is FALSE.*
~~The IUT is configured with two action lists that include a sequence of externally visible outputs with a write somewhere in the sequence that will fail. The action lists are identical except that one has Quit_On_Failure set to TRUE and the other set to FALSE. The TD triggers both action lists. The target propertyexternal outputs are observed to verify that the failure procedures are properly implemented. If the IUT does not support action list configuration for this Test Concept, then this test shall be omitted.~~

Configuration Requirements: The IUT shall be configured with a Command object having at least two action lists, X and Y, each of which is configured with at least two discernable property/value write operations such that X(P1, V1) <> X(P2,V2)

and Y(P3,V3) <> Y(P4,V4). The property/value write operations, X(P1,V1) and Y(P3,V3) will fail. Quit_On_Failure for X has a value of TRUE, Quit_On_Failure for Y has a value of FALSE. At the start of the test, each target property is configured with a value different than what the action will attempt to write and Present_Value of O1 <> X.
~~that includes writing to a sequence of externally visible outputs. Somewhere in the sequence there shall be a write command that will fail that is followed by write commands that will succeed. Both action lists shall be identical except that list X shall have Quit_On_Failure set to TRUE and Y shall have Quit_On_Failure set to FALSE. Delays shall be configured into the Action, if necessary, sufficient to let the tester observe that In_Process equals TRUE while the action is in process. If no Command object can be so configured, then verification of In_Process being TRUE shall be skipped~~

Test Steps:

1. READ IV1 = P1 (Initial value of P1)
2. READ IV2 = P2 (Initial value of P2)
3. WRITE Present_Value = *the array index for* X
4. WAIT ***Internal Processing Fail Time***
5. WHILE (In_Process = TRUE) {}
~~2. VERIFY In_Process = TRUE~~
~~3. CHECK (for the externally visible actions and verify that they stop when the failure occurs)~~
-- (for any externally visible actions, verify that they stop when the failure occurs)
~~7. VERIFY In_Process = FALSE,~~
~~5~~*6.* VERIFY All_Writes_Successful = FALSE
7. VERIFY P1 = IV1
8. VERIFY P2 = IV2
9. READ IV3 = P3 (Initial value of P3)
10. READ IV4 = P4 (Initial value of P4)
11. WRITE Present_Value = *the array index for* Y
12. WAIT ***Internal Processing Fail Time***
~~7. VERIFY In_Process = TRUE~~
~~8. CHECK (for the externally visible actions and verify that they continue to the end after the failure occurs)~~
-- (for any externally visible actions, verify that they continue to the end after the failure occurs)
13. WHILE (In_Process = TRUE) {}
~~14. VERIFY In_Process = FALSE,~~
14. VERIFY All_Writes_Successful = FALSE
15. VERIFY P3 = IV3
16. VERIFY P4 = V4 -- V4 is applied because Quit_On_Failure is false.

7.3.2.9.4 Empty Action List Test

Reason for Change: Remove reference to externally visible output.

Purpose: To verify that a Command object takes no action when Present_Value is written to with a non-zero value that corresponds to an empty action list.

Test Concept: The IUT is configured with at least one empty action list. The TD triggers the action list. The external outputs are observed to verify that no changes occurred. If the IUT does not support action list configuration for this Test Concept, then this test shall be omitted.

Configuration Requirements: The IUT shall be configured with a Command object that has an Action property with at least one empty action list.

Test Steps:

1. WRITE Present_Value = (an index corresponding to an empty action list)
~~2. RECEIVE BACnet-SimpleACK-PDU~~
2. VERIFY In_Process = FALSE
3. VERIFY All_Writes_Successful = TRUE
~~5. CHECK (if any of the actions of the Command object have externally visible results verify that no changes occurred)~~
-- (if any of the actions of the Command object have externally visible results verify that no changes occurred)

7.3.2.9.5 Action 0 Test

Reason for Change: Remove reference to externally visible output.

Purpose: To verify that a Command object takes no action when Present_Value is written to with a value of 0.

Test Concept: The Present_Value of a Command object, configured with at least one non-empty action list, is written with a value of 0. All_Writes_Successful is verified to have a value of TRUE.

Configuration Requirements: The IUT shall be configured with a Command object that has an Action property with at least one non-empty action list.

Test Steps:

1. WRITE Present_Value = 0
2. ~~RECEIVE BACnet-SimpleACK-PDU~~
2. WAIT (**Internal Processing Fail Time**)
3. VERIFY In_Process = FALSE
4. VERIFY All_Writes_Successful = TRUE
5. ~~CHECK (if any of the actions of the Command object have externally visible results verify that no changes occurred)~~
   -- (if any of the actions of the Command object have externally visible results verify that no changes occurred)

7.3.2.9.7 Write While In_Process is TRUE Test

Reason for Change: Remove reference to externally visible output.

Purpose: To verify that an action list continues to completion if a second action list is commanded while In_Process is TRUE and that the second action list is not executed.

Test Concept: The IUT is configured with two action lists that include *manipulating a property value. The TD triggers first action list and immediately triggers the second action list which must return an error. The manipulated property value is checked to ensure only the first action list was successful.* ~~a sequence of externally visible outputs with post delays for each action. The TD triggers first action list. The external outputs are observed in order to trigger the second action list during the post delay of the first list. The TD triggers the second action list. The external outputs are observed to verify that the second action list is not executed. If the IUT does not support Post Delay, then this test shall be omitted.~~ If the IUT does not support action list configuration for this Test Concept, then this test shall be omitted.

Configuration Requirements: The IUT shall be configured with a Command object *O1* having two distinct action lists, X and Y, that include writing to *property P1 with X writing value V1 and Y writing value V2*~~a sequence of externally visible outputs~~. *Action list X will a contain post delay, PD1, sufficient to allow the TD time to write action list Y before timing out.*~~There shall be a post delay between writes to the externally visible outputs that is long enough for the tester to observe the delay (This ensures In_Process remains TRUE long enough to command the second action list)~~.

Test Steps:

1. WRITE Present_Value = *the array index for* X
2. ~~WRITE Present_Value = Y~~
2. *BEFORE PD1*
3. *TRANSMIT WriteProperty-Request,*
   'Object Identifier' = O1,
   'Property Identifier' = Present_Value,
   'Property Value' = Y
3. ~~IF (Protocol_Revision is present and Protocol_Revision > 10) THEN~~
4. RECEIVE BACnet-Error-PDU,
   Error Class = OBJECT,
   Error Code = BUSY
   ~~ELSE~~
5. ~~(RECEIVE BACnet-Error-PDU,~~
   ~~Error Class = OBJECT,~~
   ~~Error Code = BUSY) |~~

~~6. (RECEIVE BACnet-Error-PDU~~
~~Error Class = SERVICES,~~
~~Error Code = SERVICE_REQUEST_DENIED | OTHER)~~
~~4. CHECK (that the externally visible actions of X take place)~~
~~5. CHECK (that the externally visible actions of Y do not take place)~~
5. WAIT (PD1)
6. VERIFY P1 = V1
7. VERIFY In_Process = FALSE,
8. VERIFY All_Writes_Successful = TRUE

7.3.2.39.11 Transition Test

Reason for Change: Remove checking physical output.

Purpose: To verify that the Lighting Output object transitions using the configured function and transitions at the configured speed when Transition is set to either FADE or RAMP.

Test Concept: Setup a Lighting Output object, O1, to use fading or ramping as the default transition method. Present_Value is changed to V1 which is larger than the initial Present_Value, V0, so that the output will fade or ramp up. Halfway through the process, verify that Tracking_Value is approximately equal to the value halfway between V0 and V1. ~~The physical output shall also be verified that it is fading or ramping from V0 to V1.~~ When the process completes, verify that Tracking_Value reached V1. Repeat the process fading or ramping down from V1 to V2.

Configuration Requirements: O1 shall be configured such that all slots in the Priority_Array numerically less than PTY1 have a value of NULL and no internal algorithms are issuing commands to O1 at a priority numerically less than or equal to PTY1. The Transition property is set to FADE or RAMP, Present_Value is V0 and In_Progress is IDLE.

To test FADE functionality, T is FADE, A is FADE_ACTIVE, W1 and W2 are (Default_Fade_Time / 2), and Default_Fade_Time is sufficiently large so as to allow the intermediate progress checks.

To Test RAMP functionality, T is RAMP, A is RAMP_ACTIVE, W1 is (((V1 – V0) / Default_Ramp_Rate) / 2), W2 is (((V1 – V2) / Default_Ramp_Rate) / 2), and Default_Ramp_Rate is sufficiently small so as to allow the intermediate progress checks.

Test Steps:
1. VERIFY Transition = T

2. VERIFY In_Progress = IDLE

3. V0 = READ Present_Value

4. WRITE Present_Value = V1, ARRAY INDEX = PTY1

5. VERIFY Present_Value = V1

6. WAIT W1

7. VERIFY Tracking_Value ~= (V1 + V0) / 2
8. VERIFY In_Progress = A
~~9. CHECK (the physical output is fading from V0 to V1)~~
-- (the physical output is fading from V0 to V1)

9. WAIT W1
10 VERIFY In_Progress = IDLE
11 VERIFY Tracking_Value = V1
12 WRITE Present_Value = V2, ARRAY INDEX = PTY1
13 VERIFY Present_Value = V2
14 WAIT W2
15 VERIFY Tracking_Value ~= (V2 + V1) / 2
16 VERIFY In_Progress = A
~~18. CHECK (the physical output is fading V1 to V2)~~
-- (the physical output is fading V1 to V2)

17 WAIT W2
18 VERIFY In_Progress = IDLE
19 VERIFY Tracking_Value = V2

7.3.2.40.6 Door_Unlock_Delay_Time Test

Reason for Change: Remove checking physical output.

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

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

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

Test Steps:

-- Test PULSE_UNLOCK
1. WRITE Present_Value = PULSE_UNLOCK
2. WAIT (**Internal Processing Fail Time**)
3. BEFORE Door_Unlock_Delay_Time
4. IF (Lock_Status is present) THEN
5. VERIFY Lock_Status = LOCKED
~~6. CHECK (that the door control output is in a state that would cause the door to be locked)~~
-- door control output is in a state that would cause the door to be locked
6. IF (Lock_Status is present) THEN
7. VERIFY Lock_Status = UNLOCKED
~~9. CHECK (that the door control output is in a state that would cause the door to be unlocked)~~
-- door control output is in a state that would cause the door to be unlocked
8. WAIT (Door_Pulse_Time)
9. VERIFY Present_Value = LOCK
10. IF (Lock_Status is present) THEN
11. VERIFY Lock_Status = LOCKED
~~14. CHECK (that the door control output is in a state that would cause the door to be locked)~~
-- door control output is in a state that would cause the door to be locked)

-- Test EXTENDED_PULSE_UNLOCK
12. WRITE Present_Value = EXTENDED_PULSE_UNLOCK
13. WAIT (**Internal Processing Fail Time**)
14. BEFORE Door_Unlock_Delay_Time
15. IF (Lock_Status is present) THEN
16. VERIFY Lock_Status = LOCKED
~~20. CHECK (that the door control output is in a state that would cause the door to be locked)~~
-- door control output is in a state that would cause the door to be locked

17. IF (Lock_Status is present) THEN
18. VERIFY Lock_Status = UNLOCKED
~~19. CHECK (that the door control output is in a state that would cause the door to be unlocked)~~
-- door control output is in a state that would cause the door to be unlocked
19. WAIT (Door_Extended_Pulse_Time)
20. VERIFY Present_Value = LOCK
21. IF (Lock_Status is present) THEN

22 VERIFY Lock_Status = LOCKED
~~28. CHECK (that the door control output is in a state that would cause the door to be locked)~~
-- door control output is in a state that would cause the door to be locked

-- Test UNLOCK
23. WRITE Present_Value = UNLOCK
24. WAIT (**Internal Processing Fail Time**)
24. IF (Lock_Status is present) THEN
25. VERIFY Lock_Status = UNLOCKED
~~33. CHECK (that the door control output is in a state that would cause the door to be unlocked)~~
-- door control output is in a state that would cause the door to be unlocked

New Test

[Add this new test to BTL Specified Tests]

7.3.2.40.X1 Present_Value and Lock_Status Test

Reason for Change: No test exists for this functionality.

Purpose: To verify that writing to the Present_Value will cause an appropriate change to Lock_Status.

Test Concept: The Present_Value property is written with each of the following values: UNLOCK, LOCK, PULSE_UNLOCK, EXTENDED_PULSE_UNLOCK and the Lock_Status property is monitored.

Configuration Requirements: The Relinquish_Default shall have the value LOCK. All writes are at a priority higher than any internal algorithms writing to this property. Out_Of_Service shall be set to FALSE. Prior to the test the Present_Value shall have the value LOCK and the IUT is in a state that would cause the door to be locked. Door_Pulse_Time and Door_Extended_Pulse_Time shall be configured with values large enough to allow verification of Lock_Status when the door is in an unlocked state.

Test Steps:

-- *Test UNLOCK value*
1. WRITE Present_Value = UNLOCK
2. WAIT (**Internal Processing Fail Time**)
3. VERIFY Lock_Status = UNLOCKED

-- *Test LOCK value*
4. WRITE Present_Value = LOCK
5. WAIT (**Internal Processing Fail Time**)
6. VERIFY Lock_Status = LOCKED

-- *Test PULSE_UNLOCK value*
7. WRITE Present_Value = PULSE_UNLOCK
8. WAIT (**Internal Processing Fail Time**)
9. WAIT (Door_Unlock_Delay_Time if present)
10. VERIFY Lock_Status = UNLOCKED
11. WAIT (Door_Pulse_Time)
12. VERIFY Present_Value = LOCK
13. VERIFY Lock_Status = LOCKED

-- *Test EXTENDED_PULSE_UNLOCK value*
14. WRITE Present_Value = EXTENDED_PULSE_UNLOCK
15. WAIT (**Internal Processing Fail Time**)
16. WAIT (Door_Unlock_Delay_Time if present)
17. VERIFY Lock_Status = UNLOCKED
18. WAIT (Door_Extended_Pulse_Time)

19. VERIFY Present_Value = LOCK
20. VERIFY Lock_Status = LOCKED

Removed Tests (Reference)

[Remove the following tests from 135.1]

7.3.2.6.3 Polarity Property Tests

7.3.2.40.1 Commandable Present_Value Test

7.3.2.40.3 Door_Status with Physical Door Status Tests

7.3.2.40.4 Lock_Status Tests

BTL-26.1 imp3-12: Clarify the Restart_Notification_Recipients Test [BTLWG-457]

Overview:

8.20.1 Base Requirements
Base requirements must be met by any IUT claiming conformance to this BIBB.
135.1-2025 - 8.3.10 - Device Restart Notifications
...
The testing hints should be moved to Test Directives and changed to something like:
Repeat with a unicast recipient, if the property is not readonly with a local broadcast recipient.
Repeat with a broadcast recipient.

135, Clause 12.11.42 states: " The default value of the property shall be a single entry representing a broadcast on the local network. If the property is not writable, then it shall contain the default value. " The Restart_Notification_Recipients property cannot be configurable.

Changes:

Checklist Changes

None

Test Plan Changes

[Modify the Test Directives and Testing Hints as shown below]
8.20 Device Management - Restart - B
8.20.1 Base Requirements
Base requirements must be met by any IUT claiming conformance to this BIBB.

***BTL~~135.1-2025~~* - 8.3.10 - Device Restart Notifications**		
	Test Conditionality	Must be executed.
	Test Directives	If the Restart_Notification_Recipients is writable, repeat the test with Restart_Notification_Recipients containing local broadcast, global broadcast, unicast with a BACnetAddress and unicast with a device id. Ensure that a recipient with the address form having a network number of 0 is not used when the IUT is configured as a BACnet router.
	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).~~
...		

Specified Test Changes

[Move test 8.3.10 into BTL Specified Tests - if necessary, and modify]

[8.3.10 was modified in https://btlworkinggrp.atlassian.net/browse/BTLWG-1833 . The yellow highlighting is the 1833 work item and the green highlighting is the changes from this work item.]

8.3.10 Device Restart Notifications

Purpose: To verify that the IUT initiates UnconfirmedCOVNotification service requests to each entry in its Restart_Notification_Recipients property when it resets.

Test Concept: The IUT is configured to send restart notifications and is then reset. The TD checks for the restart notifications.

Device restart notifications differ from subscribed COV notifications that use the UnconfirmedCOVNotification service in two respects. First, subscription is made through the Restart_Notification_Recipients property instead of SubscribeCOV. Second, the 'Subscriber Process Identifier' parameter always has a value of zero.

Configuration Requirements: For each Recipient of the Restart_Notification_Recipients property in the IUT which is of the device form, there shall be a device on the network that will answer Who-Is requests so that the IUT can determine addressing information before sending the restart notification.

Notes to tester: Not all IUTs can accurately differentiate between the types of restart reasons and thus no requirements are placed on the value returned in the restart notification(s). The test shall pass regardless of the order in which the restart notifications are sent to the recipients. If the Restart_Notification_Recipients list has multiple recipients, then the Time_Of_Device_Restart value is expected to be the same in all notifications resulting from the same restart. *Time_Of_Device_Restart must be a specified date and time, however the value is a local matter if time is unknown on restart.*

Test Steps:

1. IF (Restart_Notification_Recipients is writable) THEN
2. WRITE(Restart_Notification_Recipients = any non-empty list of Recipients)
   ELSE
~~3. MAKE (Restart_Notification_Recipients contain any non-empty list of Recipients)~~
3. VERIFY (Restart_Notification_Recipients = local network broadcast)
4. MAKE(the IUT reset)
5. REPEAT X = (each entry in the Restart_Notification_Recipients) DO {
6. BEFORE **Notification Fail Time**
7. RECEIVE UnconfirmedCOVNotification-Request,
   DESTINATION = X,
   'Subscriber Process Identifier' = 0,
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = (the IUT Device Identifier),
   'Time Remaining' = 0,
   'List of Values' = (System_Status=OPERATIONAL,
   Time_Of_Device_Restart = (T~~2~~),
   Last_Restart_Reason=(any valid restart reason, R))
   }
8. VERIFY Time_Of_Device_Restart = T~~2~~
9. VERIFY Last_Restart_Reason = R
10. VERIFY Local_Time ~= T.time
11. VERIFY Local_Date = T.date
12. VERIFY System_Status = OPERATIONAL
~~6. IF (T2 is not a sequence number) THEN~~
~~VERIFY Local_Time ~= T2~~
~~ELSE~~
~~MAKE(the IUT reset)~~
~~REPEAT X = (each entry in the Restart_Notification_Recipients) DO {~~
~~BEFORE **Notification Fail Time**~~
~~RECEIVE UnconfirmedCOVNotification-Request,~~
~~DESTINATION = X,~~
~~'Subscriber Process Identifier' = 0,~~
~~'Initiating Device Identifier' = IUT,~~
~~'Monitored Object Identifier' = (the IUT Device Identifier),~~
~~'Time Remaining' = 0,~~
~~'List of Values' = (System_Status=OPERATIONAL,~~
~~Time_Of_Device_Restart = (T3),~~
~~Last_Restart_Reason=(any valid restart reason, R))~~
~~7. CHECK (T3 > T2)~~

BTL-26.1 imp3-13: Simplify the EPICS Consistency Tests [BTLWG-724]

Overview:

This checking is already completely done in 5-i so 5-m is just duplicate effort without additional value. it can therefore be deleted.

Changes:

Checklist Changes

None

Test Plan Changes

~~135.1-2025~~ BTL - 5 - EPICS Consistency Tests		
	Test Conditionality	Must be executed
	Test Directives	The EPICS and the IUT Configuration must match exactly. The EPICS must note the existence of every object and property, resolution, and range restrictions if any for writable values. All writable standard properties must be claimed.
	Testing Hints	
...		

Specified Test Changes

[Move EPICS Consistency Tests from 135.1 into BTL Specified Tests and modify as shown below]

5. EPICS CONSISTENCY TESTS

Reason for change: This checking is already completely done in 5-i so 5-m is just duplicate effort without additional value. it can therefore be deleted.

These tests are static tests of the EPICS and do not involve interrogating the IUT. There are no Configuration Requirements or Test Step sections with TCSL in these tests because the tests are static tests of the EPICS and not tests of the IUT itself. Each implementation shall be tested to ensure consistency among interrelated data elements. These tests shall include:

(a) All object types required by the specified BIBBs shall be indicated as supported in the Standard Object Types Supported section of the EPICS.

(b) A minimum of one instance of each object type required by the specified BIBBs shall be included in the test database.

(c) The Protocol_Object_Types_Supported property of the Device object in the test database shall indicate support for each object type required by the supported BIBBs.

(d) All application services required by the supported BIBBs shall be indicated as supported in the BACnet Standard Application Services Supported section of the EPICS with Initiate and Execute indicated as required by the supported BIBBs.

(e) The Protocol_Services_Supported property of the Device object in the test database shall indicate support for each application service for which the supported BIBBs requires support for execution of the service.

(f) The object types listed in the Standard Object Types Supported section of the EPICS shall have a one-to-one correspondence with object types listed in the Protocol_Object_Types_Supported property of the Device object contained in the test database. An object type is supported if it can be made to exist in the IUT`s database.

(g) For each object type listed in the Standard Object Types Supported* there shall be at least one object of that type in the test database. It is permissible for there to be no instance of the File object type if File objects are dynamically creatable and come into existence only temporarily during Backup and Restore. * An object type is supported if it can be made to exist in the IUT's database.

(h) There shall be a one-to-one correspondence between the objects listed in the Object_List property of the Device object and the objects included in the test database. The Object_List property and the test database shall both include all proprietary objects. Properties of proprietary objects that are not required by BACnet Clause 23.4.3 need not be included in the test database.

(i) For each object included in the test database, all required properties for that object as defined in Clause 12 of BACnet shall be present. Standard properties which are not defined for the implemented Protocol_Revision shall not be present. In addition, if any of the properties supported for an object require the conditional presence of other properties, their presence shall be verified.

(j) For each property that is required to be writable, or conditionally writable, that property shall be marked as writable or conditionally writable, in the EPICS.

(k) The length of the Protocol_Services_Supported bitstring shall have the number of bits defined for BACnetProtocolServicesSupported for the IUT's declared protocol revision.

(l) The length of the Protocol_Object_Types_Supported bitstring shall have the number of bits defined for BACnetObjectTypesSupported for the IUT's declared protocol revision.

(m) ~~For each object included in the test database, any properties that are deprecated or removed shall not appear after the Protocol_Revision in which the property was deprecated or removed.~~ *This item has been removed.*

(n) If the Protocol_Revision property is present in the Device object and its value is greater than or equal to 14, the Property_List property of each object included in the test database shall have one entry for each property present, including non-standard properties with the exception of Object_Type, Object_Identifier, Object_Name, and Property_List.

(o) If the Segmentation_Supported property in the Device object is SEGMENTED_BOTH or SEGMENTED_RECEIVE, then the value of the Max_Segments_Accepted property of the Device object shall be greater than 1.

(p) For each property that is required to be read-only, that property shall not be marked as writable, or conditionality writable, in the EPICS.

(q) For each property for which the standard limits the value range, the value for the property in the EPICS, if provided, shall be within the allowable range as defined in the property definition.

(r) For each property which has a restricted value range defined in the EPICS, the restricted value range shall be within the allowable value range and contain the minimal value range as defined by the standard.

BTL-26.1 imp3-14: Fix Handling of Proprietary Property in Test 9.20.1.7 [BTLWG-1832]

Overview:

The issue is that '135.1-2025 9.20.1.7 Reading ALL Properties' limits the errors that proprietary properties may return.

Test 9.20.1.7 reads all properties from an object and verifies that either a value is returned or a READ_ACCESS_DENIED error. Since it reads all properties, there may be some proprietary properties tested.

Interpretation request IC 135-2020-19 states: "When a proprietary property is read from a device using a ReadPropertyMultiple or ReadProperty service, the device is not limited to the list of error codes defined in 15.7.1.3.1 and 15.5.1.3.1 respectively and is allowed to return an appropriate error code as defined in Clause 18.3".

Changes:

Checklist Changes

None

Test Plan Changes

[Move test 9.20.1.7 from 135.1-2025 into BTL Specified Tests]

Specified Test Changes

[Move test 9.20.1.7 into BTL Specified Tests, and modify]

9.20.1.7 Reading ALL Properties

Reason for change: Modify 'Notes to Tester' to clarify acceptable proprietary property errors. Remove mentions of PR 7. Add test numbers.

Purpose: To verify the ability to correctly execute a ReadPropertyMultiple service request that uses the special property identifier ALL. One instance of each object-type supported is tested.

Notes to Tester: If a property which is not readable using the ReadPropertyMultiple service is in the specified object, ~~and Protocol_Revision < 7, then either no entry is returned, or an error code is returned. If Protocol_Revision >= 7,~~ then the entry shall contain 'Error Class': PROPERTY and 'Error-Code': READ_ACCESS_DENIED for ~~that property.~~ *standard properties, and relevant errors from clause 18.3 of the BACnet Standard for proprietary properties.* Property_List (371) shall not appear in the List of Results.

Test Steps:

1. REPEAT ObjectX = (one instance of each supported object type) DO {
2. TRANSMIT ReadPropertyMultiple-Request,
   'Object Identifier' = ObjectX,
   'Property Identifier' = ALL
3. RECEIVE ReadPropertyMultiple-ACK,
   'Object Identifier' = ObjectX,
   'List Of Results' = (a list of all standard properties
   documented for ObjectX in the EPICS plus all proprietary
   properties present in ObjectX, each with a valid
   value, excluding the Property_List property)
   }