BACnet® TESTING LABORATORIES

INTERIM TEST SPECIFICATION

To Be Used with Test Package 15.2
Version 17
January 22, 2019

Approved by the BTL Working Group on January 10, 2019
Approved by the BTL Working Group Voting Members on February 6, 2019.
Published on February 11, 2019.

Foreward

The purpose of this document is to define interim tests and other test package changes made to support testing of a device that supports functionality currently not covered in the released BTL Test Package. This document should be applied and used with BTL Test Package 15.2

Vendors who are planning to submit a device for testing and who implement Protocol_Revision 16 and higher, or which contain functionality not covered by the Official Test Package, should use this Interim Test document.

Please note that there may be other tests for other functional areas that may also be required for your device. Please contact the BTL Manager before submitting your device for testing to ensure you are aware of all tests that will need to be applied to your device.

The changes in this document are for interim use only and may or may not be used as documented here when the final changes are applied to the next Test Package revision. Devices tested using this interim test document shall be recalled for updated testing when the next revision of test package is released that includes the topics covered here.

The changes in this document are summarized below:

BTL-TP15.0-0.1.0 Tests for the Network Port object (Protocol_Revision 17 or higher)
BTL-TP15.0-0.2.0 Tests for the Elevator Group object (Protocol_Revision 18 or higher)
BTL-TP15.0-0.3.0 Tests for the Escalator object (Protocol_Revision 18 or higher)
BTL-TP15.0-0.4.0 Tests for the Lift object (Protocol_Revision 18 or higher)
BTL-TP15.0-0.5.0 Network Port OPTIONAL properties clarified (Protocol_Revision 18 or higher)
BTL-TP15.0-0.6.0 Test of Write-BDT-NAK to Write-BDT service (Protocol_Revision 17 or higher)
BTL-TP15.0-0.7.0 Tests for the claim of NM-BBMDC-B (Protocol_Revision 18 or higher)
BTL-TP15.0-1.1.0 Tests for the FAULT_LISTED algorithm (Protocol_Revision 18 or higher)
BTL-TP15.0-1.2.0 Tests for FAULT-to-FAULT transitions in FAULT_LISTED algorithm (Protocol_Revision 18 or higher)
BTL-TP15.0-2.1.0 Binary Lighting Output object (Protocol_Revision 16 or higher)
BTL-TP15.1-2.2.0 Binary Lighting Output object for DS-COV-A (Protocol_Revision 16 or higher)
BTL-TP15.1-2.3.0 Binary Lighting Output object for DS-COV-B (Protocol_Revision 16 or higher)
BTL-TP15.1-2.4.0 Binary Lighting Output object for DM-OCD-B (Protocol_Revision 16 or higher)
BTL-TP15.0-3.1.0 NM-CE-A Test Considerations (Protocol_Revision 2 or higher)
BTL-TP15.0-4.1.0 Read-only Recipient_List Test Considerations (Protocol_Revision 13 or higher)
BTL-TP15.0-4.2.0 Tests for the claim of AE-CRL-B (Protocol_Revision 2 or higher)
BTL-TP15.0-5.1.0 Tests for the Lighting Output object (Protocol_Revision 14 or higher)
BTL-TP15.1-5.2.0 Lighting Output object for DS-COV-B (Protocol_Revision 14 or higher)
BTL-TP15.0-6.1.0 Tests for the claim of DS-COVP-B (Protocol_Revision 2 or higher)
BTL-TP15.0-7.1.0 Tests for the claim of NM-FDR-A (Protocol_Revision 2 or higher)
BTL-TP15.0-8.1.0 Tests for the claim of GW-EO-B (Protocol_Revision 2 or higher)
BTL-TP15.0-9.1.0 Tests for the Life Safety Point object (Protocol_Revision 2 or higher)
BTL-TP15.0-9.2.0 Tests for the Life Safety Zone object (Protocol_Revision 2 or higher)
BTL-TP15.0-9.3.0 Tests for the claim of AE-LS-A (Protocol_Revision 2 or higher)
BTL-TP15.0-9.4.0 Tests for the claim of AE-LS-B (Protocol_Revision 2 or higher)
BTL-TP15.1-0.1.0 File object (Protocol_Revision 2 or higher)
BTL-TP15.2-0.1.0 Load Control object (Protocol_Revision 6 or higher)
BTL-TP15.2-1.1.0 Access Door object (Protocol_Revision 6 or higher)

In the following document, language to be added to existing clauses of ANSI/ASHRAE 135.1-2013 or any part of the Test Package 15.2 are indicated through the use of *italics*, while deletions are indicated by ~~strikethrough~~. Where entirely new sections are proposed to be added, plain type is used throughout.

Table of Contents

BTL-TP15.0-0.1.0 Tests for the Network Port object

A device including a Network Port object must claim Protocol_Revision 17 or higher and comply with the following section.

[In BTL Checklist, add new Network Port section in existing 3. Object testing.]

Support	**Listing**	**Option**
Network Port Object		
	R	Base Requirements
	S	Supports writable Out_Of_Service properties

[In BTL Test Plan, add new Network Port section to 3. Object testing]

3.X43 Network Port Object

3.X43.1 Base Requirements

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

BTL - 7.3.2.X43.1 - Network Port ACTIVATE_CHANGES test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.2.X43.2 - Network Port non-volatility properties test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 9.18.X5 - ReadProperty of the Network Port Object using the Unknown Instance		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X43.2 Supports writable Out_Of_Service properties

The Out_Of_Service property in Network Port objects contained in the IUT is either writable or can be modified by any other means.

<table>
<tr><td colspan="3"><b>BTL - 7.3.2.X43.3 - Out_Of_Service, Status_Flags, and Reliability test for an Object that does not contain Present_Value</b></td></tr>
<tr><td></td><td><b>Test Method</b></td><td>Manual</td></tr>
<tr><td></td><td><b>Configuration</b></td><td>As per <b><i>BTL Specified Tests</i></b>.</td></tr>
<tr><td></td><td><b>Test Conditionality</b></td><td>If this property is writable, this test must be executed.</td></tr>
<tr><td></td><td><b>Test Directives</b></td><td>This test shall be applied to a Network Port object.</td></tr>
<tr><td></td><td><b>Testing Hints</b></td><td></td></tr>
<tr><td></td><td><b>Notes &amp; Results</b></td><td></td></tr>
</table>

[In BTL Specified Tests, add three new tests 7.3.2.X43.X1 through 7.3.2.X43.X3, and one ReadProperty positive service test 9.18.1.X5 as indicated.]

7.3.2.X43.1 Network Port ACTIVATE_CHANGES test

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

Purpose: This test verifies that after any of the Network Port properties are changed, the revised value is activated when executing a ReinitializeDevice ACTIVATE_CHANGES service request.

Test Concept: Write any of the writable properties of a Network Port object, and activate those changes by issuing a ReinitializeDevice – WARMSTART or ACTIVATE_CHANGES service request. Then after the IUT has time to have finished its update, verify that the Network Port object properties contain the values that were written.

Test Steps:

```
1.  WRITE (any writable Network Port property) = (a value different from current value)
2.  VERIFY  Changes_Pending = TRUE
3.  TRANSMIT ReinitializeDevice-Request
            'Reinitialized State of Device' = WARMSTART | ACTIVATE_CHANGES
            'Password' = (any valid password)
4.  RECEIVE BACnet-SimpleACK-PDU
5.  CHECK (that the IUT has had time to have finished its update)
6.  REPEAT X - for each changed Network Port property)
        VERIFY X = (the revised value to which it was changed)
7.  VERIFY Changes_Pending = FALSE
```

7.3.2.X43.2 Network Port non-volatility properties test

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

Purpose: This test verifies that after any of the Network Port properties is changed, and the revised value is activated, then the revised value with which it was configured is maintained through a power failure and device restart.

Test Concept: Write any of the writable properties of a Network Port object (multiple properties may be written), and activate those changes by issuing a ReinitializeDevice – WARMSTART or ACTIVATE_CHANGES service request. Then after the IUT has time to have finished its update, restart the IUT device by temporarily removing power. When the device has resumed operation after that restart, verify that the Network Port object properties contain the values that were changed and activated.

Test Steps:

```
1.  WRITE (X, any writable Network Port property) = (a value different from current value)
2.  TRANSMIT ReinitializeDevice-Request
            'Reinitialized State of Device' = WARMSTART | ACTIVATE_CHANGES
            'Password' = (any valid password)
3.  RECEIVE BACnet-SimpleACK-PDU
```

```
4.  WAIT for IUT to have finished its update
5.  CHECK (that the IUT has had time to have finished its update)
6.  VERIFY X = (the revised value to which it was changed)
7.  MAKE (the IUT power cycle to reinitialize)
8.  VERIFY X = (the revised value to which it was changed)
```

7.3.2.X43.3 Out_Of_Service, Status_Flags, and Reliability test for an Object that does not contain Present_Value

Purpose: This test verifies the interrelationship between the Out_Of_Service, Status_Flags, and Reliability properties. If the PICS indicates that the Out_Of_Service property of the object under test is not writable, and if the value of the property cannot be changed by other means, then this test shall be omitted. This test applies to objects that do not contain Present_Value.

Test Concept: Write to and verify the interrelationship between the Out_Of_Service, Status_Flags, and Reliability properties of an object which does not contain Present_Value.

Configuration Requirements: The selected object is configured such that its Reliability is NO_FAULT_DETECTED before execution of this test.

Test Steps:

```
1.  IF (Out_Of_Service is writable) THEN
        WRITE Out_Of_Service = TRUE
    ELSE
        MAKE (Out_Of_Service = TRUE)
2.  VERIFY Out_Of_Service = TRUE
3.  VERIFY Status_Flags = (?, FALSE, ?, TRUE)
4.  IF (Reliability is present and writable) THEN
        REPEAT X = (all values of the Reliability enumeration appropriate to the object type except
                        NO_FAULT_DETECTED) DO {
            WRITE Reliability = X
            VERIFY Reliability = X
            VERIFY Status_Flags = (TRUE, TRUE,?, TRUE)
            WRITE Reliability = NO_FAULT_DETECTED
            VERIFY Reliability = NO_FAULT_DETECTED
            VERIFY Status_Flags = (? FALSE, ?, TRUE)
            }
5.  CHECK (all communication of the protocol modeled by the object, through that port is disabled)
6.  IF (Out_Of_Service is writable) THEN
        WRITE Out_Of_Service = FALSE
    ELSE
        MAKE (Out_Of_Service = FALSE)
7.  VERIFY Out_Of_Service = FALSE
8.  VERIFY Status_Flags = ( ?, ?, ?, FALSE)
```

9.18.1.X5 ReadProperty of the Network Port Object using the Unknown Instance

Purpose: Verify that the IUT selects the correct object when it receives Network Port with special object instance of 4194303.

Test Concept: Execute a Read service request specifying ‘Object Identifier’ = (Network Port, 4194303). The responding BACnet-user shall treat the Object Identifier as if it correctly matched the local Network Port object representing the network port through which the request was received.

Configuration Requirements: Let X be the instance numbers of Network Port object (can be same or different objects) for the IUT. If the Protocol_Revision claimed is less than 17, this test shall be skipped.

Test Steps:

1. TRANSMIT ReadProperty-Request,
   'Object Identifier' = (Network Port, 4194303),
   'Property Identifier' = Object-Identifier
2. RECEIVE ReadProperty-ACK,
   'Object Identifier' = (Network Port, X),
   'Property Identifier' = Object-Identifier,
   'Property Value' = (Network Port, X)
3. TRANSMIT ReadProperty-Request through the same port as above,
   'Object Identifier' = (Network Port, 4194303),
   'Property Identifier' = (P: any valid property which is present in the same local Network Port object as above)
4. RECEIVE ReadProperty-ACK,
   'Object Identifier' = (Network Port, X),
   'Property Identifier' = P,
   'Property Value' = (value of P from the local Network Port object representing the network port through which the request was received)

Passing Result: The IUT shall respond as indicated conveying the value from a local Network Port object representing the network port through which the request was received.

BTL-TP15.0-0.2.0 Tests for the Elevator Group object

A device including an Elevator Group object must claim Protocol_Revision 18 or higher and comply with the following section.

[In BTL Checklist, add new Elevator Group section in existing 3.]

Support	**Listing**	**Option**
Elevator Group		
	R	Base Requirements
	R	Supports Group_Members property
	O	Supports Landing_Call_Control property

[In BTL Test Plan, add new Elevator Group section at end of existing 3. Object testing, with sections 3.X45.1 Base Requirements, and two other 3.X45.2 through 3.X45.3 sections as indicated.]

3.X45 Elevator Group Object

3.X45.1 Base Requirements

The object contains Machine_Room_ID Property.

BTL - 7.3.2.X45.1.1 - Machine_Room_ID property linking with the Positive_Integer_Value Object

	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X45.2 Supports Group_Members Property

The object contains a Group_Members Property.

BTL - 7.3.2.X45.1.2 - Linking of Lift Objects under Group_Members property of the Elevator Group Object

	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed if IUT supports Lift object.
	Test Directives	
	Testing Hints	
	Notes & Results	

BTL - 7.3.2.X45.1.3 - Linking of Escalator Objects under Group_Members property of the Elevator Group Object

	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed if IUT supports Escalator object.
	Test Directives	
	Testing Hints	

	Notes & Results	

3.X45.3 Supports Landing_Call_Control Property

The object contains a Landing_Call_Control Property.

BTL - 7.3.2.X45.1.4 - Linking of Landing_Call_Control Property Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

[Add in BTL Specified Tests, these four new tests]

7.3.2.X45.1.1 Machine_Room_ID property linking with the Positive_Integer_Value Object

Purpose: To verify that Machine_Room_ID property of Elevator Group reference the Positive_Integer_Value (PIV) object, whose Present_Value property contains the identification number for the machine room that contains the group of Lifts or Escalators, represented by this object.

Test Concept: A machine room contains the Elevator Group which is having a group of Lifts or Escalators. This machine room is mapped to the Present_Value property of Positive_Integer_Value Object which in turn is referenced to the Machine_Room_ID property of Elevator Group.

Configuration Requirements: The Machine room contains Elevator Group (EG1). OBJECT is any valid object type. X is any valid instance number in the range 0 to 4194302.

Test Steps:

```
1. IF (Machine_Room_ID contains room identification number) THEN
           VERIFY (EG1), Machine_Room_ID = (PIV, X)
   ELSE
           VERIFY (EG1), Machine_Room_ID = (OBJECT, 4194303)
```

7.3.2.X45.1.2 Linking of Lift Objects under Group_Members property of the Elevator Group Object

Purpose: This test verifies that the Group_Members property of the Elevator Group object contains the object identifier of the Lift object representing lifts contained within the group represented by this Elevator Group object.

Test Concept: Tester selects an Elevator Group and reads the Group_Members property of the Elevator Group and verifies that all the Lifts that are configured under one group are present under the Group_Members property of the Elevator Group object.

Configuration Requirements: Configure 2 Lifts (L1 and L2) under the Elevator Group (EG1).

Test Steps:

```
1. VERIFY (EG1), Group_Members = (L1, L2)
```

7.3.2.X45.1.3 Linking of Escalator Objects under Group_Members property of the Elevator Group Object

Purpose: This test verifies that the Group_Members property of the Elevator Group object contains the object identifier of the Escalator object representing the escalators contained within the group represented by this Elevator Group object.

Test Concept: Tester selects an Elevator Group and reads the Group_Members property of the Elevator Group and verifies that all the Escalators that are configured under one group are present under the Group_Members property of the Elevator Group object.

Configuration Requirements: Configure 2 Escalators (E1 and E2) under the Elevator Group (EG1).

Test Steps:

1. VERIFY (EG1), Group_Members = (E1, E2)

7.3.2.X45.1.4 Linking of Landing_Call_Control Property Test

Purpose: To verify that writing Landing_Call_Control property of Elevator Group assigns an active call to the Lift Object linked by pushing it to the Assigned_Landing_Calls property of the Lift object.

Test Concept: An Elevator Group is available, and it contains at least one Lift object. Landing_Call_Control property of the Elevator Group is written with a Floor number and direction or destination for the lift. Value written to Landing_Call_Control property is updated in the Landing_Calls property of the Elevator Group which in turn updates the Assigned_Landing_Calls property of Lift. This test shall be skipped in the event of absence of Landing_Call_Control property. If any of the Landing_Calls or Assigned_Landing_Calls property is not present, then the test steps for that specific property shall be skipped.

Configuration Requirements: The Lift (L1) should be present in the Group_Members property of Elevator Group (EG1). Lowest universal floor number of the lift < A < Highest universal floor number of the lift. Lowest universal floor number of the lift <= X <= Highest universal floor number of the lift. B = (UP | DOWN | UP_AND_DOWN) and C = (B | UP_AND_DOWN).

Test Steps:

1. WRITE (EG1), Landing_Call_Control = (Floor Number A, Direction B | Destination X)
2. VERIFY (EG1), Landing_Call_Control = (Floor Number A, Direction B | Destination X)
3. VERIFY (EG1), Landing_Calls = (Floor Number A, Direction C | Destination X)
4. VERIFY (L1), Assigned_Landing_Calls = (Floor Number A, Direction C)

Notes to Tester: Landing_Calls property may contain other entries from same lift or different lifts connected under same Elevator Group. If the Elevator Group contains more than 1 lift, value written to Landing_Call_Control may get assigned to any other lift, based on the lift algorithm.

BTL-TP15.0-0.3.0 Tests for the Escalator object

A device including an Escalator object must claim Protocol_Revision 18 or higher and must comply with the following section.

[In BTL Checklist, add new Escalator section in existing 3. Object testing.]

Support	**Listing**	**Option**
Escalator Object		
	R	Base Requirements
	S	Supports writable Out_Of_Service properties
	S	Supports Escalator_Mode property
	O	Supports Energy_Meter_Ref property
	O	Supports CHANGE_OF_STATE event algorithm with Passenger_Alarm property
	O	Supports Reliability_Evaluation_Inhibit property

[In BTL Test Plan, add new Escalator section at end of existing 3. Object testing, with Base Requirements, and five other 3.X46.2 through 3.X46.6 sections as indicated.]

3.X46 Escalator Object

3.X46.1 Base Requirements

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

BTL - 7.3.2.X46.1.1 Elevator_Group property of Escalator Object linking with Group_Members property of Elevator Group Object		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X46.2 Supports writable Out_Of_Service properties

The Out_Of_Service property in Escalator objects contained in the IUT is either writable or can be modified by any other means.

BTL - 7.3.2.X43.3 - Out_Of_Service, Status_Flags, and Reliability test for an Object that does not contain Present_Value		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If this property is writable, this test must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.2.X46.1.2 - Energy_Meter, Power_Mode and Operation_Direction Tracking Test		
	Test Method	

	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	This test must be executed if Energy_Meter or Power_Mode properties are present.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.2.X46.1.3 - Passenger_Alarm and Fault_Signals Tracking Test		
	Test Method	
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.2.X46.1.4 - Escalator_Mode Tracking Test		
	Test Method	
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	This test must be executed if Escalator_Mode property is present.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X46.3 Supports Escalator_Mode Property

The Escalator_Mode property in at least one Escalator object is present.

BTL - 7.3.2.X46.1.5 - Operation_Direction Tracks Escalator_Mode Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality Must be executed.	
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X46.4 Supports Energy_Meter_Ref Property

The Energy_Meter_Ref property in at least one Escalator object is present.

BTL - 7.3.2.X46.1.6 - Energy_Meter_Ref Property Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	This test must be executed if both Energy_Meter_Ref and Energy_Meter properties are present.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X46.5 Supports CHANGE_OF_STATE event algorithm with Passenger_Alarm property

Intrinsic event algorithm is supported using Passenger_Alarm property in at least one Escalator.

BTL - 7.3.2.X46.1.7 - CHANGE_OF_STATE for Passenger_Alarm (ConfirmedEventNotification)		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	This test must be executed if the object under test supports CHANGE_OF_STATE event algorithm with Passenger_Alarm property writable or can be modified by any other means.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.2.X46.1.8 - CHANGE_OF_STATE for Passenger_Alarm (UnconfirmedEventNotification)		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	This test must be executed if the object under test supports CHANGE_OF_STATE event algorithm with Passenger_Alarm property writable or can be modified by any other means.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X46.6 Supports Reliability_Evaluation_Inhibit Property

The IUT contains, or can be made to contain, a Reliability_Evaluation_Inhibit property that is configurable to a value of TRUE.

BTL - 7.3.1.X8.1 - Reliability_Evaluation_Inhibit Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If no object exists in the IUT for which fault conditions can be generated, then this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.1.X8.2 - Reliability_Evaluation_Inhibit Summarization Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If no object exists in the IUT for which fault conditions can be generated, then this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	

[In BTL Specified Tests, add eight new tests 7.3.2.X46.1.1 through 7.3.2.X46.1.8 as indicated.]

7.3.2.X46.1.1 Elevator_Group property of Escalator Object linking with Group_Members property of Elevator Group Object

Purpose: This test verifies that Elevator_Group property of Escalator object shall have reference of Elevator Group object whose Group_Members property contains a reference of Escalator object.

Test Concept: Escalator object falls under one specific Elevator Group object. The reference of Elevator Group object should be mentioned in Elevator_Group property of Escalator object. If there is no such Elevator Group object, Elevator_Group property shall contain an object instance of 4194303.

Configuration Requirements: The Escalator (E1), should be present under Elevator Group (EG1). OBJECT is any valid object type.

Test Steps:

```
1.  VERIFY (E1), Elevator_Group = (EG1)
2.  VERIFY (EG1), Group_Members = ((E1),….., En)
3.  IF (IUT does not contain reference of any Elevator Group Object) THEN
        VERIFY (E1), Elevator_Group = (OBJECT, 4194303)
```

7.3.2.X46.1.2 Energy_Meter, Power_Mode and Operation_Direction Tracking Test

Purpose: To verify that when Out_Of_Service property is set to TRUE for the monitored Escalator object, it does not track the changes made for Energy_Meter, Power_Mode and Operation_Direction property and it does not control the escalator operation from these properties.

Test Concept: When the Out_Of_Service is set to TRUE, writing Energy_Meter, Power_Mode and Operation_Direction property shall not make escalator to update its energy value, power mode and operation direction. Also, while making escalator's energy, power mode and operation direction change from current status, it shall not get updated to Energy_Meter, Power_Mode and Operation_Direction property of the Escalator object. Out_Of_Service property of the Escalator object is set to TRUE in the beginning of the test. If either of the Energy_Meter or Power_Mode properties are not present, then the test steps for that specific property shall be skipped.

Configuration Requirements: The Escalator Object supports Energy_Meter and/or Power_Mode properties. Escalator Power_Mode is TRUE and Operation_Direction is STOPPED. Escalator is having energy meter value = X. Tester shall select any value for energy meter Y; Y < 99999 or permitted by IUT. Tester shall select any Operation_Direction supported by IUT while testing.

Test Steps:

```
1.  IF (Out_Of_Service is writable) THEN
        WRITE Out_Of_Service = TRUE
    ELSE
        MAKE (Out_Of_Service = TRUE)
2.  VERIFY Out_Of_Service = TRUE
3.  VERIFY Status_Flags = (?, ?, ?, TRUE)
4.  WRITE Energy_Meter = Y
5.  VERIFY Energy_Meter = Y
6.  CHECK (the escalator's energy consumption is having value = X or value other than Y)
7.  MAKE (the escalator's energy consumption value = Z)
8.  VERIFY Energy_Meter = Y
9.  WRITE Power_Mode = FALSE
10. VERIFY Power_Mode = FALSE
11. CHECK (the escalator is still powered up independent of the value written)
12. MAKE (the escalator's power mode to be TRUE from FALSE)
13. VERIFY Power_Mode = FALSE
14. WRITE Operation_Direction = UP_RATED_SPEED
15. VERIFY Operation_Direction = UP_RATED_SPEED
```

```
16. CHECK (the escalator remains stopped)
17. MAKE (the escalator’s operation direction to be DOWN_RATED_SPEED)
18. VERIFY Operation_Direction = UP_RATED_SPEED
19. IF (Out_Of_Service is writable) THEN
        WRITE Out_Of_Service = FALSE
    ELSE
        MAKE (Out_Of_Service = FALSE)
20. VERIFY Out_Of_Service = FALSE
21. VERIFY Status_Flags = (?, ?, ?, FALSE)
```

7.3.2.X46.1.3 Passenger_Alarm and Fault_Signals Tracking Test

Purpose: To verify that when Out_Of_Service property is set to TRUE for the monitored Escalator object, it does not track the changes made for Passenger_Alarm and Fault_Signals property and it does not control the escalator operation from these properties.

Test Concept: When the Out_Of_Service is set to TRUE, writing Passenger_Alarm and Fault_Signals property shall not make escalator to update its alarm and fault status. Also, while making escalator’s fault and alarm status change from current value, it shall not get updated to Passenger_Alarm and Fault_Signals property of the Escalator object. Out_Of_Service property of the Escalator object is set to TRUE in the beginning of the test. If Fault_Signals property is not present, then the respective test steps shall be skipped.

Configuration Requirements: Escalator has no alarm or fault at the start of test. Tester shall select any value for Fault_Signals property testing that is supported by IUT.

Test Steps:

```
1.  IF (Out_Of_Service is writable) THEN
            WRITE Out_Of_Service = TRUE
    ELSE
            MAKE (Out_Of_Service = TRUE)
2.  VERIFY Out_Of_Service = TRUE
3.  VERIFY Status_Flags = (?, ?, ?, TRUE)
4.  WRITE Passenger_Alarm = TRUE
5.  VERIFY Passenger_Alarm = TRUE
6.  CHECK (the escalator’s alarm is not triggered)
7.  MAKE (the escalator  in NORMAL state)
8.  VERIFY Passenger_Alarm = TRUE
9.  WRITE Fault_Signals = OVERSPEED_FAULT
10. VERIFY Fault_Signals = OVERSPEED_FAULT
11. CHECK (the escalator does not have any fault into it)
12. MAKE (the escalator to have SAFETY_DEVICE_FAULT fault)
13. VERIFY Fault_Signals = OVERSPEED_FAULT
14. IF (Out_Of_Service is writable) THEN
            WRITE Out_Of_Service = FALSE
    ELSE
            MAKE (Out_Of_Service = FALSE)
15. VERIFY Out_Of_Service = FALSE
16. VERIFY Status_Flags = (?, ?, ?, FALSE)
```

7.3.2.X46.1.4 Escalator_Mode Tracking Test

Purpose: To verify that when Out_Of_Service property is set to TRUE for the monitored Escalator object, it does not track the changes made for Escalator_Mode property and also it does not control the escalator operation from this property.

Test Concept: When the Out_Of_Service is set to TRUE, writing Escalator_Mode property shall not make escalator to update its mode. Also, while making escalator’s mode to change from current value, it shall not get updated to Escalator_Mode property of the Escalator object. Out_Of_Service property of the Escalator object is set to TRUE in the beginning of the test. If this property is not present, then this test shall be skipped.

Configuration Requirements: The Escalator Object shall support Escalator_Mode property. Escalator runs at UP mode. Tester shall select any value for Escalator_Mode property for testing that are supported by IUT.

Test Steps:

```
1.  IF (Out_Of_Service is writable) THEN
        WRITE Out_Of_Service = TRUE
    ELSE
        MAKE (Out_Of_Service = TRUE)
2.  VERIFY Out_Of_Service = TRUE
3.  VERIFY Status_Flags = (?, ?, ?, TRUE)
4.  WRITE Escalator_Mode = DOWN
5.  VERIFY Escalator_Mode = DOWN
6.  CHECK (the escalator or slanted passenger conveyor is still moving upward)
7.  MAKE (the escalator to move from downward to upward)
8.  VERIFY Escalator_Mode = DOWN
9.  IF (Out_Of_Service is writable) THEN
        WRITE Out_Of_Service = FALSE
    ELSE
        MAKE (Out_Of_Service = FALSE)
10. VERIFY Out_Of_Service = FALSE
11. VERIFY Status_Flags = (?, ?, ?, FALSE)
```

7.3.2.X46.1.5 Operation_Direction Tracks Escalator_Mode Test

Purpose: To verify the linking of Operation_Direction property and Escalator_Mode property of Escalator object

Test Concept: Operation_Direction property i.e. the direction and speed in which this escalator is presently moving corresponds to the Escalator_Mode property of Escalator object

Test Steps:

```
1.  IF (Escalator_Mode = STOP) THEN
        VERIFY Operation_Direction = STOPPED
2.  IF (Escalator_Mode = UP) THEN
        VERIFY Operation_Direction = UP_RATED_SPEED | UP_REDUCED_SPEED
3.  IF (Escalator_Mode = DOWN) THEN
        VERIFY Operation_Direction = DOWN_RATED_SPEED | DOWN_REDUCED_SPEED
```

7.3.2.X46.1.6 Energy_Meter_Ref Property Test

Purpose: To verify linking of Energy_Meter property and Energy_Meter_Ref property.

Test Concept: If the Energy_Meter_Ref property is present and initialized with and Object (contains an instance other than 4194303), then the Energy_Meter property, if present, shall have a value of 0.0. If Energy_Meter_Ref property is un-initialized, then the Energy_Meter property shall have any valid value.

Test Steps:

```
1.  IF (Energy_Meter_Ref is present and initialized with instance other than 4194303) THEN
```

```
        VERIFY Energy_Meter = 0.0
    ELSE
        VERIFY Energy_Meter = (Any Valid Value)
```

7.3.2.X46.1.7 CHANGE_OF_STATE for Passenger_Alarm (ConfirmedEventNotification)

Purpose: To verify the correct operation of the CHANGE_OF_STATE event algorithm. This test applies to Event Enrollment objects with an Event_Type of CHANGE_OF_STATE and to intrinsic event reporting for Escalator and Lift objects.

Test Concept: The object begins the test in a NORMAL state. pMonitoredValue is set to TRUE. After pTimeDelay the object shall enter the OFFNORMAL state and transmit an event notification message. pMonitoredValue is set to FALSE corresponding to a NORMAL state. After pTimeDelayNormal the object shall enter the NORMAL state and transmit an event notification message

Configuration Requirements: The IUT shall be configured such that the Event_Enable property has a value of TRUE for the TO-OFFNORMAL, TO-FAULT and TO-NORMAL transitions. The Issue_Confirmed_Notifications parameter shall have a value of TRUE. The event-generating objects shall be in a NORMAL state at the start of the test. If a Notification Class object is being used to configure recipient information the value of the Transitions parameter for all recipients shall be (TRUE, TRUE, TRUE). If present in the object being tested, the Event_Detection_Enable property shall have a value of TRUE, Event_Algorithm_Inhibit shall have a value of FALSE.

Test Steps:

```
1.  VERIFY pCurrentState = NORMAL
2.  I F (the object, or referenced object, if using Event Enrollment, is an Escalator or Lift object with
        Passenger_Alarm property) THEN
3.  MAKE (pMonitoredValue (Passenger_Alarm) = TRUE)
4.  WAIT (pTimeDelay)
5.  BEFORE Notification Fail Time
        RECEIVE ConfirmedEventNotification-Request,
            'Process Identifier' =              (any valid process ID),
            'Initiating Device Identifier' =    IUT,
            'Event Object Identifier' =  (the intrinsic reporting object being tested or the EventEnrollment
            object being tested),
             'Time Stamp' =                     (T1, the current local time or sequence number),
             'Notification Class' =             (the configured notification class),
            'Priority' =        (the value configured to correspond to a TO-OFFNORMAL transition),
            'Event Type' =                      CHANGE_OF_STATE,
            'Message Text' =                    (optional, any valid message text),
            'Notify Type' =                     EVENT | ALARM,
            'AckRequired' =                     TRUE | FALSE,
            'From State' =                      NORMAL,
            'To State' =                        OFFNORMAL,
            'Event Values' =                    (pMonitoredValue, pStatusFlags)
6.  TRANSMIT BACnet-SimpleACK-PDU
7.  VERIFY pStatusFlags = (TRUE, FALSE, ?, ?)
8.  VERIFY pCurrentState = OFFNORMAL
9.  VERIFY Event_Time_Stamps = (T1, *, *)
10. MAKE (pMonitoredValue (Passenger_Alarm) = FALSE)
11. WAIT (pTimeDelayNormal)
12. BEFORE Notification Fail Time
        RECEIVE ConfirmedEventNotification-Request,
            'Process Identifier' =              (any valid process ID),
            'Initiating Device Identifier' =    IUT
```

'Event Object Identifier' =	(the intrinsic reporting object being tested or the EventEnrollment object being tested),
'Time Stamp' =	(T2, the current local time or sequence number),
'Notification Class' =	(the configured notification class),
'Priority' =	(the value configured to correspond to a TO-NORMAL transition),
'Event Type' =	CHANGE_OF_STATE,
'Message Text' =	(optional, any valid message text),
'Notify Type' =	EVENT | ALARM,
'AckRequired' =	TRUE | FALSE,
'From State' =	OFFNORMAL,
'To State' =	NORMAL,
'Event Values' =	(pMonitoredValue, pStatusFlags)

13. TRANSMIT BACnet-SimpleACK-PDU
14. VERIFY pStatusFlags = (FALSE, FALSE, ?, ?)
15. VERIFY pCurrentState = NORMAL
16. VERIFY Event_Time_Stamps = (T1, *, T2)

7.3.2.X46.1.8 CHANGE_OF_STATE for Passenger_Alarm (UnconfirmedEventNotification)

Purpose: To verify the correct operation of the CHANGE_OF_STATE event algorithm. This test applies to Event Enrollment objects with an Event_Type of CHANGE_OF_STATE and to intrinsic event reporting for Escalator and Lift objects.

Test Concept: The object begins the test in a NORMAL state. pMonitoredValue is set to TRUE. After pTimeDelay the object shall enter the OFFNORMAL state and transmit an event notification message. pMonitoredValue is set to FALSE corresponding to a NORMAL state. After pTimeDelayNormal the object shall enter the NORMAL state and transmit an event notification message

Configuration Requirements: The IUT shall be configured such that the Event_Enable property has a value of TRUE for the TO-OFFNORMAL, TO-FAULT and TO-NORMAL transitions. The Issue_Confirmed_Notifications parameter shall have a value of FALSE. The event-generating objects shall be in a NORMAL state at the start of the test. If a Notification Class object is being used to configure recipient information the value of the Transitions parameter for all recipients shall be (TRUE, TRUE, TRUE). If present in the object being tested, the Event_Detection_Enable property shall have a value of TRUE, Event_Algorithm_Inhibit shall have a value of FALSE.

Test Steps: The test steps for this test are identical to the test steps in 7.3.2.X46.1.7 except that the ConfirmedEventNotification requests are UnconfirmedEventNotification requests and the TD does not acknowledge receiving the notifications.

BTL-TP15.0-0.4.0 Tests for the Lift object

A device including a Lift object must claim Protocol_Revision 18 or higher and must comply with the following section.

[In BTL Checklist, add new Lift section in existing 3]

Support	Listing	Option
Lift Object		
	R	Base Requirements
	S	Supports writable Out_Of_Service properties
	S	Supports Landing_Door_Status and Car_Door_Status properties
	O	Supports Making_Car_Call, and Register_Car_Call properties
	O	Supports BACnetARRAY Properties related to the doors of a car
	O	Supports Car_Position and Next_Stopping_Floor properties
	O	Supports Assigned_Landing_Calls, Making_Car_Call and Registered_Car_Call properties
	O	Supports Energy_Meter_Ref and Energy_Meter properties
	O	Supports Higher_Deck and Lower_Deck properties
	O	Supports Reliability_Evaluation_Inhibit property
	O	Supports Reliability Evaluation
	O	Supports CHANGE_OF_STATE event algorithm with Passenger_Alarm property
	O	Supports writable Assigned_Landing_Calls property

[In BTL Test Plan, add new Lift section at end of existing 3. Object testing, with sections 3.X47.1 Base Requirements, and twelve other 3.X47.2 through 3.X47.13 sections as indicated.

3.X47 Lift Object

3.X47.1 Base Requirements

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

BTL - 7.3.2.X47.1.1 - Elevator_Group property of Lift Object linking with Group_Members property of Elevator Group Object.		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X47.2 Supports writable Out_Of_Service properties

The Out_Of_Service property in Lift objects contained in the IUT is either writable or can be modified by any other means.

BTL - 7.3.2.X43.3 - Out_Of_Service, Status_Flags, and Reliability test for an Object that does not contain Present_Value		
	Test Method	Manual

	Configuration	This test shall be executed using a Lift object.
	Test Conditionality	If this property is writable, this test must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.2.X47.1.2 - Car_Moving_Direction and Car_Assigned_Direction Tracking Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If Out_Of_Service property is either writable or can be modified by other means and if any of these properties are present, this test must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.2.X47.1.3 - Car_Door_Status and Landing_Door_Status Tracking Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If Out_Of_Service property is either writable or can be modified by other means and if any of these properties are present, this test must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.2.X47.1.4 - Car_Position and Next_Stopping_Floor Tracking Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If Out_Of_Service property is either writable or can be modified by other means and if any of these properties are present, this test must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.2.X47.1.5 - Passenger_Alarm and Fault_Signals Tracking Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If Out_Of_Service property is either writable or can be modified by other means and if any of these properties are present, this test must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.2.X47.1.6 - Making_Car_Call, Car_Mode & Car_Door_Command Tracking Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If Out_Of_Service property is either writable or can be modified by other means and if any of these properties are present, this test must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.2.X47.1.7 - Assigned_Landing_Call and Registered_Car_Call Tracking Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.

	Test Conditionality	If Out_Of_Service property is either writable or can be modified by other means and if any of these properties are present, this test must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.2.X47.1.8 - Car_Door_Zone and Car_Load Tracking Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If Out_Of_Service property is either writable or can be modified by other means and if any of these properties are present, this test must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.2.X47.1.9 - Energy_Meter and Car_Drive_Status Tracking Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If Out_Of_Service property is either writable or can be modified by other means and if any of these properties are present, this test must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X47.3 Supports Making_Car_Call and Register_Car_Call Properties

Either of the Making_Car_Call, Register_Car_Call properties in at least one Lift object are present.

BTL - 7.3.2.X47.1.10 - Making_Car_Call and Registered_Car_Call Tests		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	This test must be executed if Making_Car_Call and Registered_Car_Call properties are present.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X47.4 Supports BACnetARRAY Properties related to the doors of a car

BACnetARRAY properties related to the doors of a car are present in at least one Lift object.

BTL - 7.3.2.X47.1.11 - Array Size of the Lift Object properties based on car door size		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	This test must be executed if any of the BACnetARRAY properties Car_Door_Text, Assigned_Landing_Calls, Making_Car_Call, Registered_Car_Call, Car_Door_Status, Car_Door_Command and Landing_Door_Status are present.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X47.5 Supports Landing_Door_Status and Car_Door_Status Properties

The Landing_Door_Status property in at least one Lift object is present.

BTL - 7.3.2.X47.1.12 - Landing_Door_Status Tracks Car_Door_Status Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	This test must be executed if Landing_Door_Status property is present.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X47.6 Supports Car_Position and Next_Stopping_Floor Properties

Either of the Car_Position,Next_Stopping_Floor property in at least one Lift object is present.

BTL - 7.3.2.X47.1.13 - Highest Universal floor number linking to Car_Position and Next_Stopping_Floor properties		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	This test must be executed if Car_Position and Next_Stopping_Floor properties are present. If any property is not present, the respective step shall be skipped
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X47.7 Supports Assigned_Landing_Calls, Making_Car_Call and Registered_Car_Call Properties

Either of the Assigned_Landing_Calls, Making_Car_Call and Register_Car_Call property in at least one Lift object is present.

BTL - 7.3.2.X47.1.14 Highest Universal floor number linking to Assigned_Landing_Calls, Making_Car_Call and Registered_Car_Call properties		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	This test must be executed if Assigned_Landing_Calls, Making_Car_Call and Registered_Car_Call properties are present. If any property is not present, the respective step shall be skipped
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X47.8 Supports Energy_Meter_Ref and Energy_Meter Properties

The Energy_Meter_Ref and Energy_Meter property in at least one Lift object is present.

BTL - 7.3.2.X47.1.15 Energy_Meter_Ref Property Tests		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	This test must be executed if Energy_Meter_Ref and Energy_Meter property is present

	Test Directives	
	Testing Hints	
	Notes & Results	

3.X47.9 Supports Higher_Deck and Lower_Deck Properties

The Higher_Deck and Lower_Deck properties in at least one Lift object is present.

BTL - 7.3.2.X47.1.16 Higher_Deck and Lower_Deck Tests		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	This test must be executed if Higher_Deck and Lower_Deck properties are present
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X47.10 Supports Reliability_Evaluation_Inhibit Property

The IUT contains, or can be made to contain, a Reliability_Evaluation_Inhibit property that is configurable to a value of TRUE.

BTL - 7.3.1.X8.1 - Reliability_Evaluation_Inhibit Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If no object exists in the IUT for which fault conditions can be generated, then this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.1.X8.2 - Reliability_Evaluation_Inhibit Summarization Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If no object exists in the IUT for which fault conditions can be generated, then this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X47.11 Supports Reliability Evaluation

The IUT contains, or can be made to contain, a Lift object that can generate ConfirmedEventNotifications and UnconfirmedEventNotifications with an Event_Type of CHANGE_OF_RELIABILITY.

BTL - 8.4.X1.13 Change_Of_Reliability with FAULT_LISTED Algorithm (ConfirmedEventNotification)		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	This test must be executed
	Test Directives	
	Testing Hints	

	Notes & Results	
BTL - 8.4.X1.14 Change_Of_Reliability with FAULT_LISTED Algorithm (UnconfirmedEventNotification)		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	This test must be executed
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X47.12 Supports CHANGE_OF_STATE event algorithm with Passenger_Alarm property

Intrinsic event algorithm is supported using Passenger_Alarm property in at least one Lift object.

BTL - 7.3.2.X46.1.8 CHANGE_OF_STATE for Passenger_Alarm (ConfirmedEventNotification)		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	This test must be executed if the object under test supports CHANGE_OF_STATE event algorithm with Passenger_Alarm property writable or can be modified by any other means.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.2.X46.1.9 CHANGE_OF_STATE for Passenger_Alarm (UnconfirmedEventNotification)		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	This test must be executed if the object under test supports CHANGE_OF_STATE event algorithm with Passenger_Alarm property writable or can be modified by any other means.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X47.13 Supports writable Assigned_Landing_Calls Property

The Assigned_Landing_Calls property is present in at least one Lift object.

BTL - 7.3.2.X47.1.17 - Linking of Assigned_Landing_Calls property of Lift Object to Landing_Calls property of Elevator Group		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	This test must be executed if Assigned_Landing_Calls is writable.
	Test Directives	
	Testing Hints	
	Notes & Results	

[In BTL Specified Tests, add the following new tests]

7.3.2.X47.1.1 Elevator_Group property of Lift Object linking with Group_Members property of Elevator Group Object.

Purpose: This test verifies that Elevator_Group property of Lift object shall have reference of Elevator Group object whose Group_Members property contains a reference of Lift object.

Test Concept: Lift object falls under one specific Elevator Group object. The reference of Elevator Group object should be mentioned in Elevator_Group property of Lift object. If there is no such Elevator Group object, Elevator_Group property shall contain an object instance of 4194303.

Configuration Requirements: The Lift (L1) should present under the Elevator Group (EG1). OBJECT is any valid object type.

Test Steps:

```
1.  VERIFY (L1), Elevator_Group = (EG1)
2.  VERIFY (EG1), Group_Members = ((L1), .... Ln)
3.  IF (IUT does not have reference of any such Elevator Group object) THEN
        VERIFY (L1), Elevator_Group = (OBJECT, 4194303)
```

7.3.2.X47.1.2 Car_Moving_Direction and Car_Assigned_Direction Tracking Test

Purpose: To verify that when Out_Of_Service property is set to TRUE for the monitored Lift object, it does not track the changes made for Car_Moving_Direction and Car_Assigned_Direction property and it does not control the lift operation from these properties.

Test Concept: When Out_Of_Service is set to TRUE, writing Car_Moving_Direction and Car_Assigned_Direction property shall not make lift to serve specified direction. Also, making lift to serve any direction shall not be updated in Car_Moving_Direction and Car_Assigned_Direction property of Lift object. Out_Of_Service property of the Lift object is set to TRUE in the beginning of the test. If Car_Assigned_Direction property is not present, then the respective test steps shall be skipped.

Configuration Requirements: ‘X’ and ‘Y’ are any valid directions supported by IUT. Tester shall select any car moving direction and car assigned direction supported by IUT.

Test Steps:

```
1.  IF (Out_Of_Service is writable) THEN
        WRITE Out_Of_Service = TRUE
    ELSE
        MAKE (Out_Of_Service = TRUE)
2.  VERIFY Out_Of_Service = TRUE
3.  VERIFY Status_Flags = (?, ?, ?, TRUE)
4.  WRITE Car_Moving_Direction = Direction X
5.  VERIFY Car_Moving_Direction = Direction X
6.  CHECK (the lift is not serving as per the Car_Moving_Direction property)
7.  MAKE (the lift to move in Direction Y)
8.  VERIFY Car_Moving_Direction = Direction X
9.  WRITE Car_Assigned_Direction = Direction X
10. VERIFY Car_Assigned_Direction = Direction X
11. CHECK (the lift is not serving as per the Car_Assigned_Direction property)
12. MAKE (the lift assigned towards Direction Y)
13. VERIFY Car_Assigned_Direction = Direction X
14. IF (Out_Of_Service is writable) THEN
        WRITE Out_Of_Service = FALSE
    ELSE
        MAKE (Out_Of_Service = FALSE)
15. VERIFY Out_Of_Service = FALSE
16. VERIFY Status_Flags = (?, ?, ?, FALSE)
```

7.3.2.X47.1.3 Car_Door_Status and Landing_Door_Status Tracking Test

Purpose: To verify that when Out_Of_Service property is set to TRUE for the monitored Lift object, it does not track the changes made for Car_Door_Status and Landing_Door_Status property and it does not control the lift operation from these properties.

Test Concept: When Out_Of_Service is set to TRUE, writing Car_Door_Status and Landing_Door_Status property shall not make lift and landing doors to operate. Also, making lift and landing doors to operate shall not be updated in Car_Door_Status and Landing_Door_Status property when the Out_Of_Service is set to TRUE. Out_Of_Service property of the Lift object is set to TRUE in the beginning of the test. If Landing_Door_Status property is not present, then the respective test steps shall be skipped.

Configuration Requirements: Lift’s Door starts in OPEN State. ARRAY INDEX = (any valid value N; 1≤ N ≤ number of doors of a car). Universal floor number = (X = any valid floor number of the lift connected to the IUT) Tester shall select any car door status and landing door status values supported by IUT.

Test Steps:

```
1.  IF (Out_Of_Service is writable) THEN
        WRITE Out_Of_Service = TRUE
    ELSE
        MAKE (Out_Of_Service = TRUE)
2.  VERIFY Out_Of_Service = TRUE
3.  VERIFY Status_Flags = (?, ?, ?, TRUE)
4.  WRITE Car_Door_Status = CLOSED, ARRAY INDEX = N
5.  VERIFY Car_Door_Status = CLOSED, ARRAY INDEX = N
6.  CHECK (the lift’s car door is not operating as per the Car_Door_Status property)
7.  MAKE (the lift’s car door N to OPEN)
8.  VERIFY Car_Door_Status = CLOSED, ARRAY INDEX = N
9.  WRITE Landing_Door_Status = CLOSING, ARRAY INDEX = N, Universal floor number = X
10. VERIFY Landing_Door_Status = CLOSING, ARRAY INDEX = N
11. CHECK (the specified landing door is not serving as per the Landing_Door_Status property)
12. MAKE (the landing door for car door N to OPEN at Universal floor number X)
13. VERIFY Landing_Door_Status = CLOSING, ARRAY INDEX = N, Universal floor number = X
14. IF (Out_Of_Service is writable) THEN
        WRITE Out_Of_Service = FALSE
    ELSE
        MAKE (Out_Of_Service = FALSE)
15. VERIFY Out_Of_Service = FALSE
16. VERIFY Status_Flags = (?, ?, ?, FALSE)
```

7.3.2.X47.1.4 Car_Position and Next_Stopping_Floor Tracking Test

Purpose: To verify that when Out_Of_Service property is set to TRUE for the monitored Lift object, it does not track the changes made in Car_Position and Next_Stopping_Floor property and also it does not control the lift operation from these properties.

Test Concept: When the Out_Of_Service is set to TRUE, writing Car_Position and Next_Stopping_Floor property shall not make lift to update its car position and next stopping floor. Also, while making lift’s car position and next stopping floor change from current value, it shall not get updated to Car_Position and Next_Stopping_Floor property of the Lift object. Out_Of_Service property of the Lift object is set to TRUE in the beginning of the test. If Next_Stopping_Floor property is not present, then the respective test steps shall be skipped.

Configuration Requirements: Lift's current position (floor) is A. Universal floor number = (X, Y, A, B, C = any valid floor number of the lift connected to the IUT). Tester shall select any floor number supported by IUT for this test.

Test Steps:

```
1.  IF (Out_Of_Service is writable) THEN
        WRITE Out_Of_Service = TRUE
    ELSE
        MAKE (Out_Of_Service = TRUE)
2.  VERIFY Out_Of_Service = TRUE
3.  VERIFY Status_Flags = (?, ?, ?, TRUE)
4.  WRITE Car_Position = Y
5.  VERIFY Car_Position = Y
6.  CHECK (the lift still stands at the floor A)
7.  MAKE (the lift to stand at the floor X)
8.  VERIFY Car_Position = Y
9.  WRITE Next_Stopping_Floor = C
10. VERIFY Next_Stopping_Floor = C
11. CHECK (the lift is not moving towards floor C and it still stands at floor X)
12. MAKE (the lift to move from floor X to reach floor B)
13. VERIFY Next_Stopping_Floor = C
14. IF (Out_Of_Service is writable) THEN
        WRITE Out_Of_Service = FALSE
    ELSE
        MAKE (Out_Of_Service = FALSE)
15. VERIFY Out_Of_Service = FALSE
16. VERIFY Status_Flags = (?, ?, ?, FALSE)
```

7.3.2.X47.1.5 Passenger_Alarm and Fault_Signals Tracking Test

Purpose: To verify that when Out_Of_Service property is set to TRUE for the monitored Lift object, it does not track the changes made for Passenger_Alarm and Fault_Signals property and it does not control the lift operation from these properties.

Test Concept: When the Out_Of_Service is set to TRUE, writing Passenger_Alarm and Fault_Signals property shall not make lift to update its alarm and fault status. Also, while making lift's fault and alarm status change from current value, it shall not get updated to Passenger_Alarm and Fault_Signals property of the Lift object. Out_Of_Service property of the Lift object is set to TRUE in the beginning of the test. If Fault_Signals property is not present, then the respective test steps shall be skipped.

Configuration Requirements: Lift has no alarm or fault at the start of test. Tester shall select any value for Fault_Signals property testing that is supported by IUT.

Test Steps:

```
1.  IF (Out_Of_Service is writable) THEN
        WRITE Out_Of_Service = TRUE
    ELSE
        MAKE (Out_Of_Service = TRUE)
2.  VERIFY Out_Of_Service = TRUE
3.  WRITE Passenger_Alarm = TRUE
4.  VERIFY Passenger_Alarm = TRUE
5.  CHECK (the lift's alarm is not triggered)
6.  MAKE (the lift to move from Alarm to normal state)
7.  VERIFY Passenger_Alarm = TRUE
```

```
8.  WRITE Fault_Signals = CALL_BUTTON_STUCK
9.  VERIFY Fault_Signals = CALL_BUTTON_STUCK
10. CHECK (the lift does not have any fault into it)
11. MAKE (the lift to have POSITION_LOST fault)
12. VERIFY Fault_Signals = CALL_BUTTON_STUCK
13. IF (Out_Of_Service is writable) THEN
        WRITE Out_Of_Service = FALSE
    ELSE
        MAKE (Out_Of_Service = FALSE)
14. VERIFY Out_Of_Service = FALSE
```

7.3.2.X47.1.6 Making_Car_Call, Car_Mode & Car_Door_Command Tracking Test

Purpose: To verify that when Out_Of_Service property is set to TRUE for the monitored Lift object, it does not track the changes made for Making_Car_Call, Car_Mode & Car_Door_Command property and also it does not control the lift operation from these properties.

Test Concept: When Out_Of_Service is set to TRUE, writing Making_Car_Call, Car_Mode & Car_Door_Command property shall not make lift to serve specified floor, to set the mode and to execute car door commands. Also, making lift to serve different floors, to operate at different modes and for various car door commands shall not be updated in Making_Car_Call, Car_Mode & Car_Door_Command properties of Lift Object. Out_Of_Service property of the Lift object is set to TRUE in the beginning of the test. If any of the Making_Car_Call, Car_Mode or Car_Door_Command property is not present, then the test steps for that specific property shall be skipped.

Configuration Requirements: Car_Mode is NORMAL and Car_Door_Command is CLOSE at the start of the test. ARRAY INDEX = (any valid value N; 1≤ N ≤ number of doors of a car). Universal floor number = (X, Y = any valid floor number of the lift connected to the IUT). Tester shall select any car door command or car mode supported by IUT while testing.

Test Steps:

```
1.  IF (Out_Of_Service is writable) THEN
        WRITE Out_Of_Service = TRUE
    ELSE
        MAKE (Out_Of_Service = TRUE)
2.  VERIFY Out_Of_Service = TRUE
3.  VERIFY Status_Flags = (?, ?, ?, TRUE)
4.  WRITE Making_Car_Call = any valid floor X, ARRAY INDEX = N
5.  VERIFY Making_Car_Call = X, ARRAY INDEX = N
6.  CHECK (the lift is not serving as per value X in Making_Car_Call property)
7.  MAKE (the lift to serve call at floor Y for car door N)
8.  VERIFY Making_Car_Call = X, ARRAY INDEX = N
9.  WRITE Car_Door_Command = OPEN, ARRAY INDEX = N
10. VERIFY Car_Door_Command = OPEN, ARRAY INDEX = N
11. CHECK (the lift's car door N is not opening as per the Car_Door_Command property)
12. MAKE (the lift to CLOSE at the car door N from OPEN or NONE)
13. VERIFY Car_Door_Command = OPEN, ARRAY INDEX = N
14. WRITE Car_Mode = HOMING
15. VERIFY Car_Mode = HOMING
16. CHECK (the lift is not moving into HOMING mode)
17. MAKE (the lift into PARKING mode)
18. VERIFY Car_Mode = HOMING
19. IF (Out_Of_Service is writable) THEN
        WRITE Out_Of_Service = FALSE
    ELSE
        MAKE (Out_Of_Service = FALSE)
```

```
20. VERIFY Out_Of_Service = FALSE
21. VERIFY Status_Flags = (?, ?, ?, FALSE)
```

7.3.2.X47.1.7 Assigned_Landing_Call and Registered_Car_Call Tracking Test

Purpose: To verify that when Out_Of_Service property is set to TRUE for the monitored Lift object, it does not track the changes made for Assigned_Landing_Call and Registered_Car_Call property and it does not control the lift operation from these properties.

Test Concept: When Out_Of_Service is set to TRUE, writing Assigned_Landing_Call and Registered_Car_Call property shall not make lift to serve specified floors and direction. Also, making lift to serve any floors and direction shall not be updated in Assigned_Landing_Calls and Registered_Car_Call property of Lift object. . Out_Of_Service property of the Lift object is set to TRUE in the beginning of the test. If any of the Assigned_Landing_Calls and Registered_Car_Call property is not present, then the test steps for that specific property shall be skipped.

Configuration Requirements: ARRAY INDEX = (any valid value N; 1≤ N ≤ number of doors of a car). Universal floor number = (A, B, X1...n, Y1…n = any valid floor number of the lift connected to the IUT). P, Q is any valid direction supported by IUT.

Test Steps:

```
1.  IF (Out_Of_Service is writable) THEN
        WRITE Out_Of_Service = TRUE
    ELSE
        MAKE (Out_Of_Service = TRUE)
2.  VERIFY Out_Of_Service = TRUE
3.  VERIFY Status_Flags = (?, ?, ?, TRUE)
4.  WRITE Assigned_Landing_Calls = (Floor A, Direction P), ARRAY INDEX = N
5.  VERIFY Assigned_Landing_Calls = (Floor A, Direction P), ARRAY INDEX = N
6.  CHECK (the lift is not serving as per the values of Assigned_Landing_Calls property)
7.  MAKE (the lift to serve landing call at Floor B, Direction Q for car door N)
8.  VERIFY Assigned_Landing_Calls = (Floor A, Direction P), ARRAY INDEX = N
9.  WRITE Registered_Car_Call = (X1, X2, X3, X4...Xn), ARRAY INDEX = N
10. VERIFY Registered_Car_Call = (X1, X2, X3, X4...Xn), ARRAY INDEX = N
11. CHECK (the lift is not serving as per the Registered_Car_Call property)
12. MAKE (the lift to serve calls at Floor (Y1, Y2, Y3….Yn) for car door N)
13. VERIFY Registered_Car_Call = (X1, X2, X3, X4...Xn), ARRAY INDEX = N
14. IF (Out_Of_Service is writable) THEN
        WRITE Out_Of_Service = FALSE
    ELSE
        MAKE (Out_Of_Service = FALSE)
15. VERIFY Out_Of_Service = FALSE
16. VERIFY Status_Flags = (?, ?, ?, FALSE)
```

7.3.2.X47.1.8 Car_Door_Zone and Car_Load Tracking Test

Purpose: To verify that when Out_Of_Service property is set to TRUE for the monitored Lift object, it does not track the changes made for Car_Door_Zone and Car_Load property and it does not control the lift operation from these properties.

Test Concept: When the Out_Of_Service is set to TRUE, writing Car_Door_Zone and Car_Load property shall not make lift update its car door zone and its load. Also, while making lift's car to enter to a particular door zone where door opening is permitted and having a specific weight of lift car shall not get updated to Car_Door_Zone and Car_Load properties of the Lift object. Out_Of_Service property of the Lift object is set to TRUE in the beginning of the test. If any of the Car_Door_Zone and Car_Load property is not present, then the test steps for that specific property shall be skipped.

Configuration Requirements: Lift is stopped at any floor in the specified car door zone and having X units of weight. Tester shall select any weight within the permissible limit of the IUT while testing the Car_Load property.

Test Steps:

```
1.  IF (Out_Of_Service is writable) THEN
            WRITE Out_Of_Service = TRUE
    ELSE
            MAKE (Out_Of_Service = TRUE)
2.  VERIFY Out_Of_Service = TRUE
3.  VERIFY Status_Flags = (?, ?, ?, TRUE)
4.  WRITE Car_Door_Zone = FALSE
5.  VERIFY Car_Door_Zone = FALSE
6.  CHECK (the lift's car door zone remains unchanged independent of value written)
7.  MAKE (the lift's car door to door opening permitted zone)
8.  VERIFY Car_Door_Zone = FALSE
9.  WRITE Car_Load = X+1 units
10. VERIFY Car_Load = X+1 units
11. CHECK (the car load is X units)
12. MAKE (the lift car load to X+2)
13. VERIFY Car_Load = X+1 units
14. IF (Out_Of_Service is writable) THEN
            WRITE Out_Of_Service = FALSE
    ELSE
            MAKE (Out_Of_Service = FALSE)
15. VERIFY Out_Of_Service = FALSE
16. VERIFY Status_Flags = (?, ?, ?, FALSE)
```

7.3.2.X47.1.9 Energy_Meter and Car_Drive_Status Tracking Test

Purpose: To verify that when Out_Of_Service property is set to TRUE for the monitored Lift object, it does not track the changes made for Energy_Meter and Car_Drive_Status property and it does not control the lift operation from these properties.

Test Concept: When the Out_Of_Service is set to TRUE, writing Energy_Meter and Car_Drive_Status property shall not make lift to update its energy value and car drive status. Also, while making lift's energy and car drive status change from current value, it shall not get updated to Energy_Meter and Car_Drive_Status property of the Lift object. Out_Of_Service property of the Lift object is set to TRUE in the beginning of the test. If any of the Energy_Meter and Car_Drive_Status property is not present, then the test steps for that specific property shall be skipped.

Configuration Requirements: Lift is stopped at any floor, i.e. car drive status is stationary. Lift is having energy meter value = X. Tester shall select any value for energy meter Y; Y < 99999 or permitted by IUT. Tester shall select any car drive status supported by IUT.

Test Steps:

```
1.  IF (Out_Of_Service is writable) THEN
          WRITE Out_Of_Service = TRUE
    ELSE
          MAKE (Out_Of_Service = TRUE)
2.  VERIFY Out_Of_Service = TRUE
3.  VERIFY Status_Flags = (?, ?, ?, TRUE)
4.  WRITE Energy_Meter = Y
5.  VERIFY Energy_Meter = Y
```

```
6.  CHECK (the lift’s energy consumption is having value = X or value other than Y)
7.  MAKE (the lift’s energy consumption value = Z)
8.  VERIFY Energy_Meter = Y
9.  WRITE Car_Drive_Status = BRAKING
10. VERIFY Car_Drive_Status = BRAKING
11. CHECK (the lift’s car drive status is STATIONARY)
12. MAKE (the lift’s car drive status to ACCELERATE)
13. VERIFY Car_Drive_Status = BRAKING
14. IF (Out_Of_Service is writable) THEN
        WRITE Out_Of_Service = FALSE
    ELSE
        MAKE (Out_Of_Service = FALSE)
15. VERIFY Out_Of_Service = FALSE
16. VERIFY Status_Flags = (?, ?, ?, FALSE)
```

7.3.2.X47.1.10 Making_Car_Call and Registered_Car_Call Test

Purpose: To verify that the values written into Making_Car_Call property of lift object reflects in its Registered_Car_Call property at the same door side array index.

Test Concept: Making_Car_Call property of Lift (L1) object being tested is subjected for car calls provided by means of passenger requesting for car stop or by means of writing the property. The Registered_Car_Call property value at a specified array index is checked to verify that it is same as that of value provided to Making_Car_Call property.

Configuration Requirements: For below steps 'Array Index' = (any valid value N; 1≤ N ≤ number of doors of a car) and 'Property Value' = (any valid value X; X ≤ highest universal floor number of the lift)

Test Steps:

```
1.  IF (Making_Car_Call is writable) THEN
          WRITE (L1), Making_Car_Call = X, ARRAY INDEX = N
     ELSE
          MAKE (Making_Car_Call = (Value of X), ARRAY INDEX = N)
2.  VERIFY (L1), Making_Car_Call = X, ARRAY INDEX = N
3.  VERIFY (L1), Registered_Car_Call = X, ARRAY INDEX = N
```

Notes to Tester: Registered_Car_Call property may contain other additional entries.

7.3.2.X47.1.11 Array Size of the Lift Object properties based on car door size.

Purpose: To verify that the size of the Car_Door_Text, Assigned_Landing_Calls, Making_Car_Call, Registered_Car_Call, Car_Door_Status, Car_Door_Command and Landing_Door_Status array corresponds to the number of car doors present in the lift car and all are of same size.

Test Concept: Above properties will be verified for the array index 0 equals the number of car doors present in the Lift (L1). If change of car door size is possible, change and REPEAT all the steps else skip. If any of above properties are not present, then skip and proceed with the test for available properties.

Test Steps:

```
1.  VERIFY (L1), Car_Door_Text = (Number of car doors present in the Lift), ARRAY INDEX = 0
2.  VERIFY (L1), Assigned_Landing_Calls = (Number of car doors present in Lift), ARRAY INDEX = 0
3.  VERIFY (L1), Making_Car_Call = (Number of car doors present in the Lift), ARRAY INDEX = 0
4.  VERIFY (L1), Registered_Car_Call = (Number of car doors present in the Lift), ARRAY INDEX = 0
5.  VERIFY (L1), Car_Door_Status = (Number of car doors present in the Lift), ARRAY INDEX = 0
```

```
6.  VERIFY (L1), Car_Door_Command = (Number of car doors present in the Lift), ARRAY INDEX = 0
7.  VERIFY (L1), Landing_Door_Status = (Number of car doors present in the Lift), ARRAY INDEX = 0
8.  CHECK (Array index 0 of all these properties shall be same)
```

7.3.2.X47.1.12 Landing_Door_Status Tracks Car_Door_Status Test

Purpose: To verify that the status of Car_Door_Status property of lift is as same as that of the Landing_Door_Status property at a particular floor.

Test Concept: Car_Door_Status property of Lift (L1) object is subjected for different BACnetDoorStatus provided by changing the door status of real time lift connected to IUT or writing to it. The door side and floor number of the lift is considered in this case. The Landing_Door_Status property value at a specified array index (door size) for a particular floor (where lift car is currently present) is checked to verify that it is same as that of the status provided to Car_Door_Status property. If Landing_Door_Status property is not present, then this test shall be skipped.

Configuration Requirements: For below steps 'Array Index' = (any valid value N; 1≤ N ≤ number of doors of a car). Y = (any valid floor number of the lift connected to the IUT). Tester shall select any value X for Car_Door_Status supported by IUT.

Test Steps:

```
1.  IF (Car_Door_Status is writable) THEN
        WRITE (L1), Car_Door_Status = X, ARRAY INDEX = N
    ELSE
        MAKE (Car_Door_Status = (Value of X), ARRAY INDEX = N)
2.  VERIFY (L1), Car_Door_Status = X, ARRAY INDEX = N
3.  VERIFY (L1), Car_Position = Y,
4.  VERIFY (L1), Landing_Door_Status = X, ARRAY INDEX = N
5.  CHECK (Landing_Door_Status property value is X only for the Universal floor number Y)
```

7.3.2.X47.1.13 Highest Universal floor number linking to Car_Position and Next_Stopping_Floor properties

Purpose: This test verifies that the highest universal floor number of the Lift object can be the maximum value of above properties depending on the floor numbers

Test Concept: Lift Object (L1) Properties Car_Position and Next_Stopping_Floor will be written with the value of highest universal floor number and greater. If there is a physical lift or any alternate way for changing the highest universal floor number, change and REPEAT all the steps else omit. If any of the dependable properties are not writable, then skip the specific property from the test.
This test shall be skipped if Floor_Text property is not present.

Configuration Requirements: For below steps 'Property Value' = (Y = highest universal floor number of the lift connected to the IUT). If Next_Stopping_Floor property is not present, then respective steps shall be skipped.

Test Steps:

```
1.  VERIFY (L1), Floor_Text = Y, ARRAY INDEX = 0
2.  IF (Car_Position is writable) THEN
        WRITE (L1), Car_Position = Y
        VERIFY (L1), Car_Position = Y
3.  TRANSMIT WriteProperty-Request,
        'Object Identifier' = (L1),
        'Property Identifier' = Car_Position,
        'Property Value' = Y+1
4.  RECEIVE BACnet-Error-PDU,
        ‘Error Class’ = PROPERTY,
        ‘Error Code’ = VALUE_OUT_OF_RANGE
5.  IF (Next_Stopping_Floor is writable) THEN
        WRITE (L1), Next_Stopping_Floor = Y
        VERIFY (L1), Next_Stopping_Floor = Y
6.  TRANSMIT WriteProperty-Request,
        'Object Identifier' = (L1),
        'Property Identifier' = Next_Stopping_Floor,
        'Property Value' = Y+1
7.  RECEIVE BACnet-Error-PDU,
        ‘Error Class’ = PROPERTY,
        ‘Error Code’ = VALUE_OUT_OF_RANGE
```

7.3.2.X47.1.14 Highest Universal floor number linking to Assigned_Landing_Calls, Making_Car_Call and Registered_Car_Call properties

Purpose: This test verifies that the highest universal floor number of the Lift object can be the maximum value of above properties depending on the floor numbers

Test Concept: Lift Object (L1) Properties Assigned_Landing_Calls, Making_Car_Call and Registered_Car_Call will be written with the value of highest universal floor number and greater. If there is a physical lift or any alternate way for changing the highest universal floor number, change and REPEAT all the steps else omit. If any of the dependable properties are not writable, then skip the specific property from the test. This test shall be skipped if Floor_Text property is not present.

Configuration Requirements: For below steps 'Array Index' = (any valid value N; 1≤ N ≤ number of doors of a car) and 'Property Value' = (Y = highest universal floor number of the lift). If any of the dependable properties are not writable, then MAKE Out_Of_Service TRUE and then write, else skip the specific property from the test.

Test Steps:

```
1.  VERIFY (L1), Floor_Text = Y, ARRAY INDEX = 0
2.  IF (Making_Car_Call is writable) THEN
         WRITE (L1), Making_Car_Call = Y, ARRAY INDEX = N
         VERIFY (L1), Making_Car_Call = Y, ARRAY INDEX = N,
3.  TRANSMIT WriteProperty-Request,
         'Object Identifier' = (L1),
         'Property Identifier' = Making_Car_Call,
         'Property Value' = Y+1
4.  RECEIVE BACnet-Error-PDU,
         ‘Error Class’ = PROPERTY,
         ‘Error Code’ = VALUE_OUT_OF_RANGE
5.  IF (Registered_Car_Call is writable) THEN
         WRITE (L1), Registered_Car_Call = Y, ARRAY INDEX = N
6.  VERIFY (L1), Registered_Car_Call = Y, ARRAY INDEX = N,
7.  TRANSMIT WriteProperty-Request,
         'Object Identifier' = (L1),
         'Property Identifier' = Registered_Car_Call,
         'Property Value' = Y+1
8.  RECEIVE BACnet-Error-PDU,
         ‘Error Class’ = PROPERTY,
         ‘Error Code’ = VALUE_OUT_OF_RANGE
9.  IF (Assigned_Landing_Call is writable) THEN
         WRITE (L1), Assigned_Landing_Call = (Y, at Z Direction), ARRAY INDEX = N
10. VERIFY (L1), Assigned_Landing_Call = (Y, at Z Direction), ARRAY INDEX = N
11. TRANSMIT WriteProperty-Request,
         'Object Identifier' = (L1),
         'Property Identifier' = Assigned_Landing_Call,
         'Property Value' = (Y+1, at Z Direction)
12. RECEIVE BACnet-Error-PDU,
         ‘Error Class’ = PROPERTY,
         ‘Error Code’ = VALUE_OUT_OF_RANGE
```

7.3.2.X47.1.15 Energy_Meter_Ref Property Tests

Purpose: To verify linking of Energy_Meter property and Energy_Meter_Ref property.

Test Concept: If the Energy_Meter_Ref property of Lift object (L1) is present and initialized (contains an instance other than 4194303), then the Energy_Meter property, if present, shall have a value of 0.0. If Energy_Meter_Ref is present and is un-initialized, then the value of Energy_Meter property shall have any valid value.

Test Steps:

```
1.  IF (Energy_Meter_Ref is present and initialized with instance other than 4194303) THEN
        VERIFY Energy_Meter = 0.0
    ELSE
        VERIFY Energy_Meter = (Any Valid Value)
```

7.3.2.X47.1.16 Higher_Deck and Lower_Deck Tests

Purpose: To verify that the Higher_Deck and Lower_Deck property of the Lift Object is referencing the Lift object that refers the car deck above and below the car deck represented by this Lift object.

Test Concept: The IUT under test is configured to have a 3-deck lift having 3 Lift Objects. The Higher_Deck and Lower_Deck Property of the Lift object is then read to verify that it is representing the correct Lift Object instances. If there is no higher deck or lower deck, then the object instance shall be 4194303.

Configuration Requirements: The IUT under test is configured to have a 3-deck lift having 3 Lift Object instances: higher deck (L1), middle deck (L2) and lower deck (L3). If the IUT have 2 Deck lift having 2 Lift Objects, then the test steps shall be modified accordingly and executed.

Test Steps:

```
1.  VERIFY (L1), Higher_Deck = (OBJECT, 4194303),
2.  VERIFY (L1), Lower_Deck = (L2),
3.  VERIFY (L2), Higher_Deck = (L1),
4.  VERIFY (L2), Lower_Deck = (L3),
5.  VERIFY (L3), Higher_Deck = (L2),
6.  VERIFY (L3), Lower_Deck = (OBJECT, 4194303)
```

7.3.2.X47.1.17 Linking of Assigned_Landing_Calls property of Lift Object to Landing_Calls property of Elevator Group

Purpose: To verify that the Landing_Calls property of Elevator Group also represents the active calls present in the Assigned_Landing_Calls property of the Lift object.

Test Concept: An Elevator Group is available, supports Landing_Calls property, and it contains at least one Lift object within this group. Assigned_Landing_Calls property of the Lift is updated with the Floor number and direction for the lift. Landing_Calls property of the Elevator Group object shall have the value as per the Assigned_Landing_Calls represented by this Lift object. For implementations where it is not possible to write to Assigned_Landing_Calls, this test shall be skipped.

Configuration Requirements: The Lift (L1) should be present in the Group_Members property of Elevator Group (EG1). Lowest universal floor number of the lift < A < Highest universal floor number of the lift. Lowest universal floor number of the lift <= X <= Highest universal floor number of the lift. B = (UP | DOWN | UP_AND_DOWN) and C = (B | UP_AND_DOWN).

Test Steps:

```
1.  IF (Assigned_Landing_Calls is writable) THEN
        WRITE Assigned_Landing_Calls = (Floor Number A, Direction B)
2.  VERIFY (L1), Assigned_Landing_Calls = (Floor Number A, Direction B)
3.  VERIFY (EG1), Landing_Calls = (Floor Number A, Direction C | Destination X)
```

Notes to Tester: Landing_Calls property may contain other entries from same lift or different lifts connected under same Elevator Group.

BTL-TP15.0-0.5.0 Test Considerations for Network Port OPTIONAL properties clarified

A device including a Network Port object and claiming Protocol_Revision 18 or higher must comply with the following section.

[In BTL Test Plan sections, add indicated Directives to apply during the performance of existing BTL Specified tests 9.20.1.8 and 9.20.1.9]

Reason for Change: There are some properties that had Conformance code “Required” in Protocol_Revision 17 Some properties in Network Port object that had Conformance code “Required” in Protocol_Revision 17, in Protocol_Revision 18 changed their Conformance code to “Optional”. See http://www.bacnet.org/Interpretations/IC135-2016-1.pdf for details.

4.4 Data Sharing - ReadPropertyMultiple - B

4.4.1 Base Requirements

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

. . .		
BTL - 9.20.1.8 - Reading OPTIONAL Properties		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	This test must be executed
	Test Directives	Note: in Protocol_Revision 18 some of the properties indicated in Network Port object in Protocol_Revision 17 were changed from Required to Optional, and shall be returned when OPTIONAL is used with ReadPropertyMultiple. They shall not be returned when REQUIRED is used with ReadPropertyMultiple.
	Testing Hints	The pre-tester *shall* ~~should~~ apply this test to every object type. If the set of properties differs between instances of the same object type in the IUT, each form of the object type *shall* ~~should~~ be tested.
	Notes & Results	
BTL - 9.20.1.9 - Reading REQUIRED Properties		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	This test must be executed
	Test Directives	Note: in Protocol_Revision 18 some of the properties indicated in Network Port object in Protocol_Revision 17 were changed from Required to Optional, and shall be returned when OPTIONAL is used with ReadPropertyMultiple. They shall not be returned when REQUIRED is used with ReadPropertyMultiple.
	Testing Hints	The pre-tester *shall* ~~should~~ apply this test to every object type. If the set of properties differs between instances of the same object type in the IUT, each form of the object type *shall* ~~should~~ be tested.

	Notes & Results	

Excerpt of 135-2016-Errata-Summary

Errata 73) **Table 12-71**, p. 516,

The Network Port object properties Network_Number, Network_Number_Quality, and APDU_Length are only required if the protocol level is BACNET_APPLICATION.

Table 12-71. Properties of the Network Port Object Type

Property Identifier	Property Datatype	Conformance Code
. . .	. . .	. . .
Network_Number	Unsigned16	~~R[1]~~ *O*[1,1bis]
Network_Number_Quality	BACnetNetworkNumberQuality	~~R~~ *O*[1bis]
. . .	. . .	. . .
APDU_Length	Unsigned	~~R~~ *O*[1bis]
. . .	. . .	. . .

[1] Required to be writable in routers, secure devices, and any other device that requires knowledge of the network number for proper operation.

[1bis] *Required if Protocol_Level is BACNET_APPLICATION.*

[2] Shall be present if, and only if, the object supports execution of any of the values of the Command property. If present, this property shall be writable.

. . . . . .

BTL-TP15.0-0.6.0 Test of Write-BDT-NAK to Write-BDT service

The operation and manipulation of Broadcast Distribution Tables in devices claiming Protocol_Revision 17 or higher is performed through operations on a Network Port object for each supported port.

[In BTL Test Plan, add test to end of Base Requirements for BACnet/IP - Annex J - BBMD]

9.4 BACnet/IP - Annex J - BBMD

9.4.1 Base Requirements

The IUT acts, or can be made to act, as a BBMD device.

These base requirements must be met by any IUT that claims to support the Annex J BACnet/IP BBMD functionality.

. . .		
BTL - 7.3.2.X43.4 - Write-BDT service is required to return Write-BDT-NAK		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed in all devices claiming Protocol_Revision >= 17.
	Test Directives	
	Testing Hints	
	Notes & Results	

[In BTL Specified Tests, add new test]

7.3.2.X43.4 Write-BDT service is required to return Write-BDT-NAK

Reason for Change: Clause J.4.4.2 mandates a change and that all devices claiming Protocol_Revision >= 17, shall behave in this changed way.

Purpose: To verify that any IUT with Protocol_Revision claimed as 17 or higher, will return Write-Broadcast-Distribution-Table NAK to every Write-Broadcast-Distribution-Table request.

Configuration Requirements: If the Protocol_Revision claimed is less than 17, this test shall be skipped.

Test Steps:

1. TRANSMIT Write-Broadcast-Distribution-Table
2. RECEIVE BVLC-Result,
   'Result Code' = Write-Broadcast-Distribution-Table NAK

BTL-TP15.0-0.7.0 Test Considerations for the NM-BBMDC-B BIBB

Devices claiming this BIBB shall comply with the following section. This BIBB was specified in Protocol_Revision 17.

Overview:

Addendum 135-2012*al* added the NM-BBMDC-B BIBB. This document makes needed changes in the BTL Test Package to claim NM-BBMDC-B.

These changes are not contained in any SSPC proposal.

Changes:

[In BTL Checklist, add new Network Management - BACnet Broadcast Management Device Configuration -B section]

Support	**Listing**	**Option**
Network Management - BACnet Broadcast Management Device Configuration - B		
	R	Base Requirements
	R	Supports Registration by Foreign Devices
	BTL-C[1]	Executes Write-Broadcast-Distribution-Table
	C[2]	Supports configurable BBMD_Broadcast_Distribution_Table property
[1] This option is required if the IUT claims Protocol_Revision 16 or lower. [2] This option is required if the IUT claims Protocol_Revision 17 or higher.		

[In BTL Test Plan, add new Network Management - BACnet Broadcast Management Device Configuration -B sections at end of section 10]

10.X3 Network Management - BACnet Broadcast Management Device Configuration - B

These tests are designed for testing implementations of a BACnet Broadcast Management Device, including the execution of Network Layer and Application Layer commands to configure the BBMD.

10.X3.1 Base Requirements

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

BTL - 14.2.1.2 - Execute Forwarded-NPDU (Two-hop Distribution)		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	

	Notes & Results	
BTL - 14.2.2.2 - Execute Original-Broadcast-NPDU (Two-hop Distribution)		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	
	Notes & Results	
135.1-2013 - 14.2.3 - Execute Original-Unicast-NPDU		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	
	Notes & Results	
135.1-2013 - 14.5.2.2 - Original-Broadcast-NPDU Which Shall Be Forwarded (Two-hop Distribution)		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 14.7.1.2 - Broadcast Message from Directly Connected IP Subnet (Two-hop Distribution)		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 14.7.2.2 - Broadcast Message Forwarded by a Peer BBMD (Two-hop Distribution)		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	
	Notes & Results	
135.1-2013 - 14.9.3 - Original-Broadcast-NPDU		

	Test Method	Manual
	Configuration	***As per ASHRAE 135.1-2013***.
	Test Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	
	Notes & Results	

10.X3.2 Supports Registration by Foreign Devices

While configured as a BBMD, the IUT supports, or can be made to support, registration by Foreign Devices and forwards as original BACnet/IP unicasts to each, any broadcasts it processes.

BTL - 14.X10.2 - Holds at least 5 Foreign Device Registrations		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 14.X10.3 - Negative Foreign Device Registration when FD_Supported is FALSE		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
135.1-2013 - 14.6.1 - Execute Read-Foreign-Device-Table		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	
	Notes & Results	
135.1-2013 - 14.6.3.1 - Non-zero-Duration Foreign Device Table Timer Operations		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	
	Notes & Results	
135.1-2013 - 14.6.5 - Execute Delete-Foreign-Device-Table-Entry Which Should Be Rejected		

	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***
	Test Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	
	Notes & Results	
135.1-2013 - 14.6.6 - Execute Delete-Foreign-Device-Table-Entry		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 14.7.3.2 - Broadcast Message From a Foreign Device (Two-hop Distribution)		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***
	Test Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	
	Notes & Results	

10.X3.3 Executes Write-Broadcast-Distribution-Table

The IUT executes Write-Broadcast-Distribution-Table to update the configured peer BBMDs.

135.1-2013 - 14.3.1 - Execute Write-Broadcast-Distribution-Table (Table Growth)		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	
	Notes & Results	
135.1-2013 - 14.3.2 - Execute Write-Broadcast-Distribution-Table (Table Shrinkage)		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	
	Notes & Results	

BTL - 14.3.3 - Verify Broadcast Distribution Table Created from the Configuration Saved During the Previous Session		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***
	Test Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 14.X10.1 - Broadcast-Distribution-Table Holds at least 5 Entries		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

10.X3.4 Supports BBMD_Broadcast_Distribution_Table property

The IUT supports the configurable BBMD_Broadcast_Distribution_Table property in Network Port objects to configure peer BBMDs.

BTL - 14.X10.4 - BBMD_Broadcast_Distribution_Table Holds at Least 5 Entries		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.2.X43.4 - Write-BDT service is required to return Write-BDT-NAK		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed in all devices claiming Protocol_Revision >= 17.
	Test Directives	
	Testing Hints	
	Notes & Results	

[Add in BTL Specified Tests, these four new tests]

14.X10.1 - Broadcast-Distribution-Table Holds at Least 5 Entries

Reason For Change: NM-BBMDC-B specifically mandates this capacity behavior is supported by the product.

Purpose: Verify that IUT implements capacity mandated for the product by NM-BBMDC-B.

Test Concept: Fill the Broadcast_Distribution_Table with at least five distinct peer BBMDs entries (in addition to the entry containing the address of itself in the table).

Configuration Requirements: In a device claiming Protocol_Revision 16 or less, the means by which the product's Broadcast-Distribution-Table is configured is not restricted to BACnet network transmissions, and can be through the product's end-user interface.

Test Steps:

```
1.  MAKE (IUT enter mode functioning as a BBMD implementation)
2.  MAKE Broadcast_Distribution_Table = (its own entry and entries for at least 5 other BBMDs))
3.  TRANSMIT Read- Broadcast-Distribution-Table
4.  RECEIVE Read-Broadcast-Distribution-Table-Ack,
        'List of BDT Entries' = (the table as configured, in any order)
```

14.X10.2 - Holds at Least 5 Foreign Device Registrations

Reason For Change: NM-BBMDC-B specifically mandates this capacity behavior is supported by BBMDs.

Purpose: Verify that when configured to accept foreign device registrations, the IUT supports at least five simultaneous foreign device registrations.

Test Concept: The IUT is configured to support foreign device registrations. Five Register-Foreign-Device requests are sent from 5 different devices, to verify that it supports five registrations simultaneously in the FDT.

Configuration Requirements: Set BBMD_Accept_FD_Registrations in the Network Port object representing the port operating as a BBMD to TRUE. The TD will be configured to emulate 5 devices.

Test Steps:

```
1.   REPEAT X = 1 to 5 {
       TRANSMIT Register-Foreign-Device
                   SOURCE             = (device X)
                   'Time-to-Live '    = (a value longer than the length of the test)
       RECEIVE BVLC-Result,
               'Result Code' = Successful completion
}
```

14.X10.3 - Negative Foreign Device Registration when FD_Supported is FALSE

Reason For Change: The standard specifically mandates that BBMD_Accept_FD_Registrations property is writable if present in BBMDs.

Purpose: Verify that when BBMD_Accept_FD_Registrations is configured as FALSE, the BBMD will accept no more foreign device registrations.

Test Concept: The IUT is configured with BBMD_Accept_FD_Registrations property as FALSE. Then it is verified that no more Register-Foreign-Device registrations succeed, though those already in the FDT operate as normal.

Configuration Requirements: BBMD_Accept_FD_Registrations in the Network Port object representing the port is initially TRUE. If no Network Port object contains the BBMD_Accept_FD_Registrations property, then this test shall be skipped.

Test Steps:

```
1.  WRITE BBMD_Accept_FD_Registrations = FALSE
2.  TRANSMIT Register-Foreign-Device
3.  RECEIVE BVLC-Result,
            'Result Code' = Register-Foreign-Device NAK
```

14.X10.4 - BBMD_Broadcast_Distribution_Table Holds at Least 5 Entries

Reason For Change: NM-BBMDC-B specifically mandates this capacity behavior is supported by the product.

Purpose: Verify that the IUT supports at least 5 peer BBMD entries in its broadcast distribution table.

Test Concept: Fill the BBMD_Broadcast_Distribution_Table with at least five distinct peer BBMDs entries (in addition to the entry containing the address of itself in the table).

Configuration Requirements: the IUT is configured to operate as a BBMD.

Test Steps:

1. WRITE BBMD_Broadcast_Distribution_Table = (its own entry and entries for at least 5 other BBMDs)
2. MAKE (that configuration active)
3. TRANSMIT Read- Broadcast-Distribution-Table
4. RECEIVE Read-Broadcast-Distribution-Table-Ack,
   'List of BDT Entries' = (the table as configured, in any order)

BTL-TP15.0-1.1.0 Tests for the FAULT_LISTED algorithm

Devices claiming support for CHANGE_OF_RELIABILITY with FAULT_LISTED algorithm must claim Protocol_Revision 18 and comply with the following section.

Overview:

Addendum 135-2012*aq*-3 at Protocol_Revision 18 added new FAULT_LISTED algorithm to vertical transport objects that provide fault reporting, and to the Event Enrollment object.

Changes:

[In BTL Specified Tests, add a new test]

8.4.X1 CHANGE_OF_RELIABILITY Tests (ConfirmedEventNotification)

8.4.X1.13 Change_Of_Reliability with FAULT_LISTED Algorithm (ConfirmedEventNotification)

Purpose: To verify the correct operation of the FAULT_LISTED event algorithm.

Test Concept: Select a fault detecting object O1 which is configured to use the FAULT_LISTED algorithm. Ensure that no other fault conditions exist in the object. Set pMonitoredList to FV1, any value whose presence in the list property indicates a FAULT_LISTED fault condition. Verify the correct transition is generated. The fault condition is removed by setting pMonitoredList to empty, a value which indicates NO_FAULT_DETECTED and verify the correct transition is generated.

Configuration Requirements: O1 is configured to detect faults and to report those using confirmed event notifications. O1 is initially configured to have no fault conditions present, and has an Event_State of NORMAL. FV1 is a value for pMonitoredList which indicates a fault condition, and an empty pMonitoredList does not indicate a fault condition.

Test Steps:

```
1.  VERIFY pCurrentReliability = NO_FAULT_DETECTED
2.  VERIFY Event_State = NORMAL
3.  IF (pMonitoredList is writable) THEN
        WRITE pMonitoredList = FV1
    ELSE
        MAKE (pMonitoredList = FV1)
4.  BEFORE Notification Fail Time
        RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' =              (any valid process Identifier),
                'Initiating Device Identifier' =    IUT
                'Event Object Identifier' =         O1
                'Time Stamp' =                      (the current local time or sequence number),
                'Notification Class' =              (the notification class configured for O1),
                'Priority' =                        (the value configured for the transition),
                'Event Type' =                      CHANGE_OF_RELIABILITY,
                'Message Text' =                    (optional, any valid message text),
                'Notify Type' =                     ALARM | EVENT,
               'AckRequired' =                      TRUE | FALSE,
                'From State' =                      NORMAL,
                'To State' =                        FAULT,
                'Event Values' =                    ( FAULT_LISTED,
                                                      (T, T, ? ?),
                                                  (A list of valid values for properties required to be reported
                                                  for O1, and 0 or more other properties of O1)
                                                 )
5.  TRANSMIT BACnet-SimpleACK-PDU
6.  VERIFY pCurrentReliability = FAULTS_LISTED
7.  VERIFY Event_State = FAULT
8.  IF (pMonitoredList is writable) THEN
        WRITE pMonitoredList = {}
    ELSE
        MAKE (pMonitoredList = {})
9.  BEFORE Notification Fail Time
        RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' =           (any valid process Identifier),
                'Initiating Device Identifier' = IUT
                'Event Object Identifier' =      O1
                'Time Stamp' =                   (the current local time or sequence number),
                'Notification Class' =           (the notification class configured for O1),
                'Priority' =                     (the value configured for the transition),
                'Event Type' =                   CHANGE_OF_RELIABILITY,
                'Message Text' =                 (optional, any valid message text),
                'Notify Type' =                  ALARM | EVENT,
                'AckRequired' =                  TRUE | FALSE,
                'From State' =                   FAULT,
                'To State' =                     NORMAL,
                'Event Values' =                 ( NO_FAULT_DETECTED,
                                                  (F, F, ? ?),
                                                  (A list of valid values for properties required to be reported
                                                  for O1, and 0 or more other properties of O1)
                                                 )
10. TRANSMIT BACnet-SimpleACK-PDU
11. pCurrentReliability = NO_FAULT_DETECTED
12. VERIFY Event_State = NORMAL
```

[In BTL Specified Tests, add a new test in this section]

8.5.X1 CHANGE_OF_RELIABILITY Tests

8.5.X1.14 Change_Of_Reliability with FAULT_LISTED Algorithm (UnconfirmedEventNotification)

Purpose: To verify the correct operation of the FAULT_LISTED event algorithm.

Test Concept: Select a fault detecting object O1 which is configured to use the FAULT_LISTED algorithm. Ensure that no other fault conditions exist in the object. Set pMonitoredList to FV1, any value whose presence in the list property indicates a FAULT_LISTED fault condition. Verify the correct transition is generated. The fault condition is removed by setting pMonitoredList to empty which indicates NO_FAULT_DETECTED and verify the correct transition is generated.

Configuration Requirements: O1 is configured to detect faults and to report those using unconfirmed event notifications. O1 is initially configured to have no fault conditions present, and has an Event_State of NORMAL. FV1 is a value for pMonitoredList which indicates a fault condition, and an empty pMonitoredList does not indicate a fault condition.

Test Steps: The test steps for this test case are identical to the test steps in ‘Change_Of_Reliability with FAULT_LISTED Algorithm (ConfirmedEventNotification)’ except that the ConfirmedEventNotification requests are UnconfirmedEventNotification requests and the TD does not acknowledge receiving the notifications.

BTL-TP15.0-1.2.0 Tests for FAULT-to-FAULT transitions in FAULT_LISTED algorithm

Devices claiming support for FAULT-to-FAULT transitions in the FAULT_LISTED algorithm must claim support for Protocol_Revision 18 and comply with the following section.

Overview:

Addendum 135-2012*aq*-3 at Protocol_Revision 18 the added FAULT_LISTED algorithm for vertical transport objects provides for optional fault-to-fault reporting.

Changes:

[In BTL Checklist, add a new optional lineitem under Escalator section in existing 3. Object testing.]

Support	**Listing**	**Option**
Escalator Object		
		. . .
	O	Supports FAULT-to-FAULT transitions in FAULT_LISTED

[In BTL Test Plan, add an additional section under Escalator in order to optionally execute the testing in 3.X46.7 as indicated.]

3.X46 Escalator Object

3.X46.7 Supports FAULT-to-FAULT transitions in FAULT_LISTED

These requirements must be met by any IUT that can contain more than one element or different values in the Fault_Signals property in any of its Escalator objects.

BTL - 8.5.X1.15 - Change_Of_Reliability FAULT-to-FAULT transitions in FAULT_LISTED		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

[In BTL Specified Tests, add a new test in this section]

8.5.X1 CHANGE_OF_RELIABILITY Tests

8.5.X1.15 Change_Of_Reliability FAULT-to-FAULT transitions in FAULT_LISTED

Purpose: To verify the correct operation of FAULT-to-FAULT transitions in FAULT_LISTED event algorithm.

Test Concept: Select a fault detecting object O1 which is configured to use the FAULT_LISTED algorithm. Ensure that a fault condition exists in the object. Set pMonitoredList to FV1, any set of non-empty values different from the current set of values. Verify the correct transition is generated. The fault condition is removed by setting pMonitoredList to empty, a value which indicates NO_FAULT_DETECTED and verify the correct transition is generated.

Configuration Requirements: O1 is configured to detect faults and to report those using unconfirmed event notifications. O1 is initially configured to have a fault conditions present by pMonitoredList containing a non-empty value, and has an Event_State of FAULT. FV1 is a value or set of values for pMonitoredList, and which the IUT will support in the pMonitoredList value. An empty pMonitoredList does not indicate a fault condition.

Test Steps:

```
1.  VERIFY pCurrentReliability = FAULTS_LISTED
2.  VERIFY Event_State = FAULT
3.  IF  (pMonitoredList is writable) THEN
        WRITE pMonitoredList = FV1
    ELSE
        MAKE (pMonitoredList = FV1)
4.  BEFORE **Notification Fail Time**
        RECEIVE UnconfirmedEventNotification-Request,
                'Process Identifier' =              (any valid process Identifier),
                'Initiating Device Identifier' =    IUT
                'Event Object Identifier' =         O1
                'Time Stamp' =                      (the current local time or sequence number),
                'Notification Class' =              (the notification class configured for O1),
                'Priority' =                        (the value configured for the transition),
                'Event Type' =                      CHANGE_OF_RELIABILITY,
                'Message Text' =                    (optional, any valid message text),
                'Notify Type' =                     ALARM | EVENT,
                'AckRequired' =                     TRUE | FALSE,
                'From State' =                      FAULT,
                'To State' =                        FAULT,
                'Event Values' =                    ( FAULT_LISTED,
                                                      (T, T, ? ?),
                                                      (A list of valid values for properties required to be reported
                                                      for O1, and 0 or more other properties of O1)
                                                    )
5.  VERIFY pCurrentReliability = FAULTS_LISTED
6.  VERIFY Event_State = FAULT
7.  IF  (pMonitoredList is writable) THEN
        WRITE pMonitoredList = {}
    ELSE
        MAKE (pMonitoredList = {})
8.  BEFORE **Notification Fail Time**
        RECEIVE UnconfirmedEventNotification-Request,
               'Process Identifier' =               (any valid process Identifier),
                'Initiating Device Identifier' =    IUT
                'Event Object Identifier' =         O1
                'Time Stamp' =                      (the current local time or sequence number),
                'Notification Class' =              (the notification class configured for O1),
                'Priority' =                        (the value configured for the transition),
                'Event Type' =                      CHANGE_OF_RELIABILITY,
                'Message Text' =                    (optional, any valid message text),
                'Notify Type' =                     ALARM | EVENT,
                'AckRequired' =                     TRUE | FALSE,
                'From State' =                      FAULT,
                'To State' =                        NORMAL,
```

'Event Values' =	(NO_FAULT_DETECTED, (F, F, ? ?), (A list of valid values for properties required to be reported for O1, and 0 or more other properties of O1))

9. VERIFY pCurrentReliability = NO_FAULT_DETECTED
10. VERIFY Event_State = NORMAL

BTL-TP15.0-2.1.0: Binary Lighting Output object

Devices claiming support for a Binary Lighting Output object must claim support for Protocol_Revision 16 and comply with the following section.

Overview:

Addendum 135-2012*az* added the Binary Lighting Output object. This document makes needed changes in the BTL Test Package to claim Binary Lighting Output object.

These changes are not contained in any SSPC proposal.

[In BTL Checklist, add Binary Lighting Output object type to Section 3, Objects]

Support	**Listing**	**Option**
Binary Lighting Output Object		
	R	Base Requirements
	R	Supports command prioritization
	S	Supports writable Out_Of_Service properties
	O	Supports blink-warn
	O	Supports writable Polarity property
	O	Supports strike count tracking
	O	Supports elapsed active time tracking
	O	Contains an object with Reliability_Evaluation_Inhibit Property

[In BTL Test Plan, add Binary Lighting Output object tests in section 3.X41. In the following addition of new clauses of the Test Plan, these are indicated as entirely new sections verbatim, with plain text, verbatim **bold**, or verbatim ***bold-italic*** as shown.]

3.X41 Binary Lighting Output Object

3.X41.1 Base Requirements

Base requirements must be met by any IUT that can contain Binary Lighting Output objects. All requirements for this object are specified in other sections.

3.X41.2 Supports Command Prioritization

135.1-2013 - 7.3.1.2 - Relinquish Default Test		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	If no object can be made to meet the configuration requirements, this test shall be skipped.
	Test Directives	
	Testing Hints	

	Notes & Results	

135.1-2013 - 7.3.1.3 - Command Prioritization Test		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X41.3 Supports Writable Out_Of_Service Properties

The Out_Of_Service property in Binary Lighting Output objects contained in the IUT are writable.

135.1-2013 - 7.3.1.1 - Out_Of_Service, Status_Flags, and Reliability Tests		
	Test Method	Manual
	Configuration	This test shall be executed using a Binary Lighting Output object.
	Test Conditionality	If Out_Of_Service is writable, this test must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X41.4 Supports Blink-warn

The IUT supports blink-warn the Binary Output object.

BTL - 7.3.1.X.1 - Blink Warn WARN Command Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

BTL - 7.3.1.X.2 - Blink Warn WARN_OFF Command Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

BTL - 7.3.1.X.3 - Blink Warn WARN_RELINQUISH Command Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

	Notes & Results	
BTL - 7.3.1.X.4 - Blink Warn STOP Command Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.1.X.5 - Blink Warn WARN Command Failure Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test with WARN_OFF and WARN_RELINQUISH commands
	Testing Hints	
	Notes & Results	
BTL - 7.3.1.X.6 - Blink Warn WARN_OFF Command Failure Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.1.X.7 - Blink Warn WARN_RELINQUISH Command Failure Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.1.X.8 - Blink Warn WARN_OFF Command Halted Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.1.X.9 - Blink Warn WARN_RELINQUISH Command Halted Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

	Notes & Results	

3.X41.5 Supports writable Polarity property

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

135.1-2013 - 7.3.2.6.3 - Polarity Property Tests		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	Must be executed
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X41.6 Supports Strike Count Tracking

The properties of the Binary Lighting Output object that collectively tracks strike counts as required.

BTL - 7.3.2.X41.10 - Binary Lighting Output Object Strike Count Tests		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed if Strike_Count property supported.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X41.7 Supports Elapsed Active Time Tracking

The properties of binary objects that collectively track active time function as required.

BTL - 7.3.1.9 - Binary Object Elapsed Active Time Tests		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If all of the active time properties are supported, it must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X41.8 Contains an object with Reliability_Evaluation_Inhibit Property

The IUT contains, or can be made to contain, a Reliability_Evaluation_Inhibit property that is configurable to a value of TRUE.

BTL - 7.3.1.X8.1 - Reliability_Evaluation_Inhibit Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If no object exists in the IUT for which fault conditions can be generated, then this test shall be skipped.
	Test Directives	

	Testing Hints	
	Notes & Results	
BTL - 7.3.1.X8.2 - Reliability_Evaluation_Inhibit Summarization Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If no object exists in the IUT for which fault conditions can be generated, then this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	

[In BTL Specified Tests, add non-object specific tests for Blink in section 7.3.1.X, applicable to both Lighting Output or Binary Lighting Output objects.]

7.3.1.X.1 Blink Warn WARN Command Test

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)
6. VERIFY Egress_Active = FALSE
7. VERIFY Priority_ Array = V1, ARRAY_INDEX = PTY1

7.3.1.X.2 Blink Warn WARN_OFF Command Test

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_Time is a non-zero value, 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.  T1 = current local time
7.  BEFORE Internal Processing Fail Time
        CHECK (blink-warn occurred)
8.  WHILE (Egress_Active = TRUE)
            VERIFY Priority_ Array = V1, ARRAY_INDEX = PTY1
9.  T2 = current local time
10. VERIFY (T1 – T2) ~= Egress_Time +/- Internal Processing Fail Time
11. VERIFY Priority_ Array = V2, ARRAY_INDEX = PTY1
```

7.3.1.X.3 Blink Warn WARN_RELINQUISH Command Test

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 have a value of NULL, slots numerically greater than PTY1 shall be V0 and 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, Egress_Active is FALSE, and Relinquish_Default has a value, V2.

	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	ON	>= 1.0 and < V1

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. T1 = current local time
7. BEFORE **Internal Processing Fail Time**
    CHECK (blink-warn occurred)
8. WHILE (Egress_Active = TRUE)
    VERIFY Priority_ Array = V1, ARRAY_INDEX = PTY1
9. T2 = current local time
10. VERIFY (T1 – T2) ~ = Egress_Time +/- **Internal Processing Fail Time**
11. VERIFY Priority_ Array = NULL, ARRAY_INDEX = PTY1

7.3.1.X.4 Blink Warn STOP Command Test

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 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 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_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
V1	ON	>1.0

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. T1 = current local time
7. BEFORE **Internal Processing Fail Time**
    CHECK (blink-warn occurred)
8. VERIFY Egress_Active = TRUE
9. WAIT less than Egress_Time
    WRITE PROP_REF = STOP, PRIORITY = PTY1
10. T2 = current local time
11. WAIT **Internal Processing Fail Time**
12. VERIFY Egress_Active = FALSE
13. WAIT Egress_Time – (T2 – T1) + **Internal Processing Fail Time**
14. VERIFY Priority_ Array = V1, ARRAY_INDEX = PTY1

7.3.1.X.5 Blink Warn WARN Command Failure Test

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)
7. VERIFY Egress_Active = FALSE
8. VERIFY Priority_ Array = V1, ARRAY_INDEX = PTY1
9. WRITE Present_Value = NULL, PRIORITY = PTY2

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

-- Test for Blink_Warn_Enable is FALSE
16. MAKE Blink_Warn_Enable = FALSE
17. WRITE PROP_REF = C1, PRIORITY = PTY1
18. WAIT **Internal Processing Fail Time**
    CHECK (blink-warn did not occur)
19. VERIFY Egress_Active = FALSE
20. VERIFY Priority_ Array = V1, ARRAY_INDEX = PTY1

7.3.1.X.6 Blink Warn WARN_OFF Command Failure Test

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)
8. VERIFY Egress_Active = FALSE
9. VERIFY Priority_ Array = V3 , ARRAY_INDEX = PTY1
10. WRITE Present_Value = V1, PRIORITY = PTY1

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

-- Test for Blink_Warn_Enable is FALSE
18. MAKE Blink_Warn_Enable = FALSE
19. WRITE PROP_REF = C1, PRIORITY = PTY1
20. WAIT **Internal Processing Fail Time**
   CHECK (blink-warn did not occur)
21. VERIFY Egress_Active = FALSE
22. VERIFY Priority_ Array = V3, ARRAY_INDEX = PTY1

7.3.1.X.7 Blink Warn WARN_RELINQUISH Command Failure Test

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 not V0, 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, slots numerically greater than PTY1 shall be V0 and 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 and Relinquish_Default is V0.

	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 to V5	ON	>1.0

Test Steps:

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

1. VERIFY Blink_Warn_Enable = TRUE
2. VERIFY Relinquish_Default <> 0
3. VERIFY Egress_Time > 0
4. WRITE Present_Value = V1, PRIORITY = PTY1
5. VERIFY Egress_Active = FALSE
6. WRITE Present_Value = V2, PRIORITY = PTY2, a value not equal to 6 and less than PTY1
7. WRITE PROP_REF = C1, PRIORITY = PTY1
8. WAIT **Internal Processing Fail Time**
   CHECK (blink-warn did not occur)
9. VERIFY Egress_Active = FALSE
10. VERIFY Priority_ Array = NULL, ARRAY_INDEX = PTY1
11. WRITE Present_Value = NULL, PRIORITY = PTY2

-- Test for the value at the specified priority is OFF or 0.0

12. WRITE Present_Value = V6, PRIORITY = PTY1
13. WRITE PROP_REF = C1, PRIORITY = PTY1
14. WAIT **Internal Processing Fail Time**
    CHECK (blink-warn did not occur)
15. VERIFY Egress_Active = FALSE
16. VERIFY Priority_ Array = NULL, ARRAY_INDEX = PTY1

-- Test for the value at the specified priority is NULL

17. WRITE Present_Value = NULL, PRIORITY = PTY1
18. WRITE PROP_REF = C1, PRIORITY = PTY1
19. WAIT **Internal Processing Fail Time**
    CHECK (blink-warn did not occur)
20. VERIFY Egress_Active = FALSE
21. VERIFY Priority_ Array = NULL, ARRAY_INDEX = PTY1

-- Test for the value of the next highest non-NULL priority is neither OFF or 0.0

22. WRITE Present_Value = V1 PRIORITY = PTY1
23. WRITE Present_Value = V3, PRIORITY = PTY3, a value numerically greater than PTY1
24. WRITE PROP_REF = C1, PRIORITY = PTY1
25. WAIT **Internal Processing Fail Time**
    CHECK (blink-warn did not occur)
26. VERIFY Egress_Active = FALSE
27. VERIFY Priority_ Array = NULL, ARRAY_INDEX = PTY1
28. WRITE Present_Value = NULL, PRIORITY = PTY3

-- Test for the value of Relinquish_Default is neither OFF or 0.0

29. WRITE Present_Value = V1, PRIORITY = PTY1
30. WRITE Relinquish_Default = V4
31. WRITE PROP_REF = C1, PRIORITY = PTY1
32. WAIT **Internal Processing Fail Time**
    CHECK (blink-warn did not occur)
33. VERIFY Egress_Active = FALSE
34. VERIFY Priority_ Array = NULL, ARRAY_INDEX = PTY1

-- Test for Blink_Warn_Enable is FALSE
35. WRITE Relinquish_Default = V5
36. WRITE Present_Value = V1, PRIORITY = PTY1
37. WRITE Blink_Warn_Enable = FALSE
38. WRITE PROP_REFPresent_Value = C1, PRIORITY = PTY1
39. WAIT **Internal Processing Fail Time**
    CHECK (blink-warn did not occur)
40. VERIFY Egress_Active = FALSE
41. VERIFY Priority_ Array = NULL, ARRAY_INDEX = PTY1

7.3.1.X.8 Blink Warn WARN_OFF Command Halted Test

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)
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
15. WRITE PROP_REF = C1, PRIORITY = PTY1
16. BEFORE **Internal Processing Fail Time**
    CHECK (blink-warn occurred)
17. BEFORE Egress_Active = FALSE
    WRITE Present_Value = V3, PRIORITY = PTY1
18. VERIFY Egress_Active = FALSE
19. VERIFY Priority_ Array = V3, ARRAY_INDEX = PTY1

7.3.1.X.9 Blink Warn WARN_RELINQUISH Command Halted Test

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, slots numerically greater than PTY1 shall be V0 and 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 and Relinquish_Default is not V0.

	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 to V3	ON	>1.0

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)
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 = 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 = V1, PRIORITY = PTY1
12. VERIFY Blink_Warn_Enable = TRUE
13. VERIFY Egress_Time > 0
14. VERIFY Egress_Active = FALSE
15. WRITE PROP_REF = C1, PRIORITY = PTY1
16. BEFORE **Internal Processing Fail Time**

```
         CHECK (blink-warn occurred)
17.   BEFORE Egress_Active = FALSE
         WRITE Present_Value = V3, PRIORITY = PTY1
18.   VERIFY Egress_Active = FALSE
19.   VERIFY Priority_ Array = V3, ARRAY_INDEX = PTY1
```

[In BTL Specified Tests, add Binary Lighting Output object specific test 7.3.1.X41.10]

7.3.2.X41.10 Binary Lighting Output Object Strike Count Tests

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

Purpose: To verify that the properties of the Bianry Lighting Output object (O1) that tracks strike counts.
Test Concept: The Present_Value or Feedback_Value of O1 can be used as the source S1 to increment Strike_Count. S1 is transitioned from OFF to ON. The Strike_Count property is checked to verify that it has been incremented. The Strike_Count is reset and Time_Of_Strike_Count_Reset is checked to verify that it has been updated appropriately. Strike_Count is set to a non-zero value and the Time_Of_Strike_Count_Reset is unchanged.

Configuration Requirements: O1 shall be configured such that the Present_Value property is writable or another means of changing these properties shall be provided.
Test Steps:

```
1     C1 = Strike_Count
2.    MAKE (S1 transition OFF to ON)
3.    VERIFY (Strike_Count = C1 + 1)
4.    IF (Strike_Count is writable) THEN
          MAKE (Strike_Count = 0)
          VERIFY (Time_Of_Strike_Count_Reset = current local time)
5.    IF (Strike_Count is writable to a non-zero value) THEN
          MAKE (Strike_Count > 0)
          VERIFY (Time_Of_Strike_Count_Reset is unchanged)
```

BTL-TP15.1-2.2.0 Binary Lighting Output object for DS-COV-A

[In BTL Interim_Tests_15.1, add the below DS-COV-A Test Plan items]

4.9.Y Can subscribe for COV from Binary Lighting Output objects

The IUT can subscribe for, receive, and process Change of Value notifications from Binary Lighting Output objects.

135.1-2013 - 9.2.1.1 - Change of Value Notifications		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	Either 9.2.1.1 or 9.3.2 must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
135.1-2013 - 9.3.2 - Change of Value Notifications		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	Either 9.2.1.1 or 9.3.2 must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

BTL-TP15.1-2.3.0 Binary Lighting Output object for DS-COV-B

[In BTL Interim_Tests_15.1, add the below DS-COV-B Test Plan items]

4.10.Y Supports COV for Binary Lighting Output objects

The IUT supports change of value notifications for at least one object of type Binary Lighting Output.

BTL - 8.2.3 - Change of Value Notification from a Binary Object Present_Value Property		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***. The selected object must be a Binary Lighting Output object.
	Test Conditionality	This may be skipped if 8.3.3 is executed against a Binary Lighting Output object.
	Test Directives	
	Testing Hints	
	Notes & Results	

BTL - 8.2.4 - Change of Value Notification from a Binary Object Status_Flags Property		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***. The selected object must be a Binary Lighting Output object.
	Test Conditionality	This may be skipped if 8.3.4 is executed against a Binary Lighting Output object.
	Test Directives	
	Testing Hints	
	Notes & Results	

BTL - 8.3.3 - Change of Value Notification from a Binary Object Present_Value Property		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***. The selected object must be a Binary Lighting Output object.
	Test Conditionality	This may be skipped if 8.2.3 is executed against a Binary Lighting Output object.
	Test Directives	
	Testing Hints	
	Notes & Results	

BTL - 8.3.4 - Change of Value Notification from a Binary Output Object Status_Flags Property		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***. The selected object must be a Binary Lighting Output object.
	Test Conditionality	This may be skipped if 8.2.4 is executed against a Binary Lighting Output object.
	Test Directives	
	Testing Hints	
	Notes & Results	

[Into BTL Interim_Tests_15.1, further revise the below versions of two tests already in BTL Specified Tests.]

8.2.3 Change of Value Notification from a Binary Object Present_Value Property

Reason for Change: Updated the 'Configuration Requirements'. Removed extraneous SimpleACKs that appear after WRITE statements. Modified descriptive text for 'List of Values' properties.

Purpose: To verify that the IUT can initiate ConfirmedCOVNotification service requests conveying a change of the Present_Value property of Binary Input, Binary Output, ~~and~~ Binary Value*, and Binary Lighting Output* objects.

Test Concept: A subscription for COV notifications is established, using a Lifetime of L. L shall be set to a value less than 24 hours and large enough to complete the test. The Present_Value of the monitored object is changed and a notification shall be received. The Present_Value may be changed using the WriteProperty service or by another means such as changing the input signal represented by a Binary Input object. For some implementations it may be necessary to write to the Out_Of_Service property first to accomplish this task. For implementations where it is not possible to write to these properties at all the vendor shall provide an alternative trigger mechanism to accomplish this task. All of these methods are equally acceptable.

Configuration Requirements: At the beginning of the test, the Out_Of_Service property shall have a value of FALSE. *Select an object where Present_Value is not expected to change outside the tester's control or which has a writable Out_Of_Service.*

Test Steps:

REPEAT X = (one supported object of each type from the set Binary Input, Binary Output, Binary Value *and Binary Lighting Output*) DO {

```
1.  TRANSMIT SubscribeCOV-Request,
        'Subscriber Process Identifier' =       (any value > 0 chosen by the TD),
        'Monitored Object Identifier' =         X,
        'Issue Confirmed Notifications' =       TRUE,
        'Lifetime' =                            L
2.  RECEIVE BACnet-SimpleACK-PDU
3.  BEFORE **Notification Fail Time**
      RECEIVE ConfirmedCOVNotification-Request,
        'Subscriber Process Identifier' =       (the same value used in step 1),
        'Initiating Device Identifier' =        IUT,
        'Monitored Object Identifier' =         X,
        'Time Remaining' =                      (any value appropriate for the Lifetime selected),
        'List of Values' =                      (the initial Present_Value and initial Status_Flags)
4.  TRANSMIT BACnet-SimpleACK-PDU
5.  IF (Out_Of_Service is writable) THEN
        WRITE X, Out_Of_Service = TRUE
                BEFORE **Notification Fail Time**
            RECEIVE ConfirmedCOVNotification-Request,
                'Subscriber Process Identifier' =  (the same value used in step 1),
                'Initiating Device Identifier' =   IUT,
                'Monitored Object Identifier' =    X,
                'Time Remaining' =                 (any value appropriate for the Lifetime selected),
                'List of Values' =                 (~~the initial~~*ReportedPV = the current* Present_Value, ~~and~~ new Status_Flags)
        TRANSMIT BACnet-SimpleACK-PDU
6.  IF (Present_Value is now writable) THEN
        WRITE X, Present_Value = (any value that differs from ~~"initial Present_Value"~~ *ReportedPV*)
            ELSE
```

```
          MAKE (Present_Value  = any value that differs from ~~"initial Present_Value"~~ ReportedPV)
7.    BEFORE Notification Fail Time
          RECEIVE ConfirmedCOVNotification-Request,
               'Subscriber Process Identifier' =       (the same value used in step 1),
               'Initiating Device Identifier' =        IUT,
               'Monitored Object Identifier' =         X,
               'Time Remaining' =                      (any value appropriate for the Lifetime selected),
               'List of Values' =                      (the new Present_Value and new Status_Flags)
8.    TRANSMIT BACnet-SimpleACK-PDU
9.    TRANSMIT SubscribeCOV-Request,
          'Subscriber Process Identifier' =            (the same value used in step 1),
          'Monitored Object Identifier' =              X
10.   RECEIVE BACnet-SimpleACK-PDU
11.   IF (Out_Of_Service is writable) THEN
          WRITE X, Out_Of_Service = FALSE
~~          RECEIVE BACnet-SimpleACK-PDU~~
```

8.2.4 Change of Value Notification from a Binary Object Status_Flags Property

Reason for Change: Updated 'Test Concept' to include case if finite lifetime is not supported. Updated 'Configuration Requirements'.

Purpose: To verify that the IUT can initiate ConfirmedCOVNotification service requests conveying a change of the Status_Flags property of Binary Input, Binary Output, ~~and~~ Binary Value, *and Binary Lighting Output* objects.

Test Concept: A subscription for COV notifications is established, using a Lifetime of L. Removed extraneous SimpleACKs after WRITE statements. The Status_Flags property of the monitored object is then changed and a notification shall be received. The value of the Status~~-~~_Flags property can be changed by using the WriteProperty service or by another means. For some implementations writing to the Out_Of_Service property will accomplish this task. For implementations where it is not possible to write to Status_Flags or Out_Of_Service or change the Status_Flags by any other means, this test shall be skipped.
Configuration Requirements: At the beginning of the test, the Out_Of_Service property shall have a value of FALSE. *Select an object where Present_Value is not expected to change outside the tester's control or which has a writable Out_Of_Service.*

Test Steps:

REPEAT X = (one supported object of each type from the set Binary Input, Binary Output, Binary Value *and Binary Lighting Output*) DO {

```
1.    TRANSMIT SubscribeCOV-Request,
          'Subscriber Process Identifier' =            (any value > 0 chosen by the TD),
          'Monitored Object Identifier' =              X,
          'Issue Confirmed Notifications' =            TRUE,
          'Lifetime' =                                 L
2.    RECEIVE BACnet-SimpleACK-PDU
3.    BEFORE Notification Fail Time
        RECEIVE ConfirmedCOVNotification-Request,
          'Subscriber Process Identifier' =            (the same value used in step 1),
          'Initiating Device Identifier' =             IUT,
          'Monitored Object Identifier' =              X,
          'Time Remaining' =                           (any value appropriate for the Lifetime selected),
          'List of Values' =                           (the initial Present_Value and initial Status_Flags)
4.    TRANSMIT BACnet-SimpleACK-PDU
```

```
5.  WRITE X, Out_Of_Service = TRUE | WRITE X, Status_Flags = (a value that differs from initial Status_Flags)
|
    MAKE (Status_Flags = any value that differs from initial Status_Flags)
~~1.  IF (WriteProperty is used in step 5) THEN~~
~~        RECEIVE BACnet-SimpleACK-PDU~~
~~7~~6.  BEFORE **Notification Fail Time**
        RECEIVE ConfirmedCOVNotification-Request,
            'Subscriber Process Identifier' =     (the same value used in step 1),
            'Initiating Device Identifier' =      IUT,
            'Monitored Object Identifier' =       X,
            'Time Remaining' =                    (any value appropriate for the Lifetime selected),
            'List of Values' =                    (~~the initial~~the current Present_Value, and new Status_Flags)
~~8~~7.  TRANSMIT BACnet-SimpleACK-PDU
~~9~~8.  TRANSMIT SubscribeCOV-Request,
        'Subscriber Process Identifier' =         (the same value used in step 1),
        'Monitored Object Identifier' =           X
~~10~~9. RECEIVE BACnet-SimpleACK-PDU
~~11~~10    IF (Out_Of_Service was changed in step 5) THEN
        WRITE X, Out_Of_Service = FALSE
~~        RECEIVE BACnet-SimpleACK-PDU~~
```

[Into BTL Interim_Tests_15, derive the below versions of two tests from 135.1-2013 tests]

8.3.3 Change of Value Notification from a Binary Object Present_Value Property

Purpose: To verify that the IUT can initiate UnconfirmedCOVNotification service requests conveying a change of the Present_Value property of Binary Input, Binary Output, ~~and~~ Binary Value*, and Binary Lighting Output* objects.

Test Steps: The steps for this test case are identical to the test steps in 8.2.3 except that the SubscribeCOV service request in step 1 shall have a value of FALSE for the 'Issue Confirmed Notifications' parameter, all of the ConfirmedCOVNotification requests shall be UnconfirmedCOVNotification requests, and there is no acknowledgment of the unconfirmed services. The MAC address used for the notification message shall be such that the TD is one of the recipients.

8.3.4 Change of Value Notification from a Binary Object Status_Flags Property

Purpose: To verify that the IUT can initiate UnconfirmedCOVNotification service requests conveying a change of the Status_Flags property of Binary Input, Binary Output, ~~and~~ Binary Value*, and Binary Lighting Output* objects.

Test Steps: The steps for this test case are identical to the test steps in 8.2.4 except that the SubscribeCOV service request in step 1 shall have a value of FALSE for the 'Issue Confirmed Notifications' parameter, all of the ConfirmedCOVNotification requests shall be UnconfirmedCOVNotification requests, and there is no acknowledgment of the unconfirmed services. The MAC address used for the notification message shall be such that the TD is one of the recipients.

BTL-TP15.1-2.4.0 Binary Lighting Output object for DM-OCD-B

[In BTL Interim_Tests_15.1, add the below DM-OCD-B Test Plan items]

8.22.X Supports Object Creation and Deletion of the Binary Lighting Output Object

The Binary Lighting Output object can be created and deleted within the IUT. The Binary Lighting Output object that is created must be the object that can be deleted using the delete service.

135.1-2013 - 9.16.1.1 - Creating Objects by Specifying the Object Type with No Initial Values		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	Must be executed on the Binary Lighting Output Object.
	Test Directives	
	Testing Hints	
	Notes & Results	

BTL - 9.16.1.2 - Creating Objects by Specifying the Object Identifier with No Initial Values		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed on the Binary Lighting Output Object.
	Test Directives	
	Testing Hints	
	Notes & Results	

135.1-2013 - 9.17.1.1 - Successful Deletion of an Object		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	Must be executed on the Binary Lighting Output Object.
	Test Directives	
	Testing Hints	
	Notes & Results	

BTL-TP15.0-3.1.0 NM-CE-A Test Considerations

Devices claiming support for the NM-CE-A BIBB must comply with the following section. This BIBB was revised in Protocol_Revision 17.

Overview:

Addendum 135-2008*v* removed the NM-CE-A BIBB from all Device Profiles. This document makes needed changes in the BTL Test Package to claim NM-CE-A.

Changes:

[In BTL Checklist, add new Network Management - Connection Establishment - A]

Network Management - Connection Establishment - A		
	R	Base Requirements

[In BTL Test Plan, append to Section 10, Network Management]

10.X4 Network Management - Connection Establishment - A

10.X4.1 Base Requirements

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

135.1-2013 - 10.5.3.1 - Establish-Connection-To-Network		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
135.1-2013 - 10.5.3.2 - Disconnect-Connection-To-Network		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

BTL-TP15.0-4.1.0 Read-only Recipient_List Test Considerations

Device claiming a it has a read-only Recipient_List property in a Notification class object must claim Protocol_Revision 13 or higher and must comply with the following section.

[In BTL Checklist, in the **Notification Class Object** revise conformance code, and add indicated lineitem.]

Notification Class Object		
	R	Base Requirements
	BTL-R	Supports DM-DDB-A
	~~BTL-R~~*C*[1]	Supports writable Recipient_List properties
	C[1]	*Supports read-only Recipient_List properties*
[1] At least one of these options must be supported.		

[In BTL Test Plan, add a new section under Notification Class Object for Supports read-only Recipient_List Properties. Entirely new sections proposed to be added in Test Plan use verbatim **bold,** or verbatim ***bold-italic*** throughout.]

3.17 Notification Class Object

. . .

3.17.4 Supports read-only Recipient_List Properties

The IUT supports read-only Recipient_List properties.

BTL - 7.3.2.21.3.X9 Read-only Recipient_List for external Notification Forwarder objects		
	Test Method	
	Configuration	As per ***BTL Specified Tests.***
	Test Conditionality	Must be executed if the IUT does not claim support for Notification Forwarder objects.
	Test Directives	
	Testing Hints	
	Notes & Results	

In BTL Test Plan, modify existing Base Requirements section under Alarm and Event - Notification - Internal-B. Modified sections in Test Plan use yellow highlighted new material to preserve the verbatim **bold,** or verbatim ***bold-italic***.

5.2 Alarm and Event - Notification - Internal-B

5.2.1 Base Requirements

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

BTL - 7.3.1.10 - Event_Enable Tests

	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If the IUT cannot be configured to meet the configuration requirements then this test shall be skipped.
	Test Directives	If Event Enrollment objects are supported, ensure this functionality is tested on Event Enrollment objects.
	Testing Hints	The BTL will apply this to a single object. The pretester should apply it to all objects that support alarm generation.
	Notes & Results	
BTL - 7.3.1.12 - Notify_Type Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If the IUT cannot be configured to meet the 135.1-2013 configuration requirements then this test shall be skipped.
	Test Directives	If Event Enrollment objects are supported, ensure this functionality is also tested on Event Enrollment objects.
	Testing Hints	
	Notes & Results	
135.1-2009 - 8.4 - ConfirmedEventNotification Service Initiation Tests		
	Test Method	
	Configuration	
	Test Conditionality	Must be executed unless IUT contains only read-only Recipient_List properties and does not claim Notification Forwarder objects. Any of the 8.4 tests can be used to ensure that the IUT properly generates ConfirmedEventNotification requests. The specific tests that can be executed are detailed under the test cases for the specific algorithms. As long as one of the tests is executed using ConfirmedEventNotifications, then this test case shall be satisfied.
	Test Directives	
	Testing Hints	
	Notes & Results	
135.1-2009 - 8.5 - UnconfirmedEventNotification Service Initiation Tests		
	Test Method	
	Configuration	
	Test Conditionality	Must be executed. Any of the 8.5 tests can be used to ensure that the IUT properly generates UnconfirmedEventNotification requests. The specific tests that can be executed are detailed under the test cases for the specific algorithms. As long as one of the tests is executed using UnconfirmedEventNotifications, then this test case shall be satisfied.
	Test Directives	
	Testing Hints	
	Notes & Results	

[In BTL Specified Tests, revise the Test Concepts for Recipient_List tests, for special situations where Recipient_List is read-only or static.]

7.3.2.21.3.1 ValidDays Test

…

Test Concept: The TD will select one instance of the Notification Class object and one instance of an event-generating object that is linked to it. The Recipient_List of the Notification Class object shall contain a single recipient with the Valid Days parameter configured so that at least one day is TRUE and at least one day is FALSE. The properties of the event-generating object will be manipulated to cause the Event_State to change from NORMAL to OFFNORMAL. The tester verifies that if the local date is one of the valid days a notification message is transmitted and the if local date is not a valid day then no notification message is transmitted. *For devices that implement a read-only Recipient_List property for all instances of Notification Class objects and are exclusively configured for all days (Valid Days set to all Days), this test shall be skipped. For devices that implement a writeable Recipient_List property for all instances of Notification Class objects, and exclusively accept all days as the only permitted configuration, this test shall be skipped.*

7.3.2.21.3.2 FromTime and ToTime Test

…

Test Concept: The case where the local date and time fall within the window defined by the From Time and To Time parameters is covered by the ValidDays test in 7.3.2.21.3.1. This test uses the same IUT configuration and sets the local time to a value that is one of the ValidDays but outside of the window defined by the From Time and To Time parameters. The objective is to verify that an event notification message is not transmitted when the event is triggered. *For devices that implement a read-only Recipient_List property for all instances of Notification Class objects and are exclusively configured for all times (From Time set to 00:00:00.0, To_Time set to 23:59:59.90), this test shall be skipped. If all instances of writeable Notification Class Recipient_List properties exclusively accept all times as the only permitted configuration, this test shall be skipped.*

7.3.2.21.3.3 IssueConfirmedNotifications Test

…

Purpose: To verify that ConfirmedEventNotification messages are used if the Issue Confirmed Notifications parameter has the value TRUE and UnconfirmedEventNotification messages are used if the value is FALSE. If the IUT does not support both confirmed and unconfirmed event notifications this test may be *skipped* ~~omitted~~. *For devices that implement a read-only Recipient_List property for all instances of Notification Class objects, and there is a value of FALSE for* the *IssueConfirmedNotifications component in all instances, this test shall be skipped.*

7.3.2.21.3.4 Transitions Test

…

Test Concept: The IUT is configured such that the Transitions parameter indicates that some event transitions are to trigger an event notification and some are not. Each event transition is triggered and the IUT is monitored to verify that notification messages are transmitted only for those transitions for which the Transitions parameter has a value of TRUE. *For devices that implement a read-only Recipient_List property for all instances of Notification Class objects and are exclusively configured for all transitions (all bits in Transitions set to TRUE), this test shall be skipped. If all instances of writeable Notification Class Recipient_List properties exclusively accept all transitions as the only permitted configuration, this test shall be skipped.*

7.3.2.21.3.5 Recipient_List Property Supports Device Identifier Recipients Test

Purpose: To verify that the Recipient_List property of the Notification Class object supports the device form of the Recipient component and that the IUT is able to associate a MAC address with the Device Identifier. The intent is to ensure that the IUT is able to locate the specified alarm recipient and send notification to the specified recipient. This test shall be run if the IUT's Notification Class object's Recipient_List property supports the BACnet object identifier form of BACnetRecipient.

Test Concept: The tester shall select a single event-generating object E in the IUT that references Notification Class object N. The tester shall add an entry into the Recipient_List of the associated Notification Class object that specifies a Device Identifier, D, for a device that the IUT is not already aware of. The TD, acting as device D, shall be located on a different network than the IUT to ensure that the IUT is capable of binding to recipients located on any network. *For devices that implement a read-only Recipient_List property for all instances of Notification Class objects and there is an address form of the Recipient component in all instances, this test shall be skipped.*

Configuration Requirements:The TD shall be configured so that it does not execute WhoHas.

Test Steps:

```
1.  WRITE N.RecipientList = ( {all days, all times, D, any process ID, FALSE, all transitions} )
2.  MAKE (the event generating object, E, transition)
3.  BEFORE Notification Fail Time  plus the amount of time the IUT takes to perform device discovery
        RECEIVE UnconfirmedEventNotification-Request,
            'Process Identifier' =              (the valid process ID from step 1),
            'Initiating Device Identifier' =    IUT,
            'Event Object Identifier' =         E,
            'Time Stamp' =                      (any valid time stamp),
            'Notification Class' =              (N's instance),
            'Priority' =                        (any valid priority),
            'Event Type' =                      (any valid event type),
            'Notify Type' =                     ALARM | EVENT,
            'AckRequired' =                     TRUE | FALSE,
            'From State' =                      (any valid event state),
            'To State' =                        (any valid event state),
            'Event Values' =                    (values appropriate to the event type)
```

Notes to Tester: The IUT is expected to initiate one or more range-restricted WhoIs requests after the modification of the Recipient_List but before the sending of the notification. The IUT might also need to perform other network discovery operations. Given that there are multiple approaches to the use of WhoIs for device discovery, the test only focuses on the IUT's ability to find device D and not on the specifics or timing of the WhoIs requests.

7.3.2.21.3.6 Recipient_List Property Supports Network Address Recipients

Purpose: To verify that the Recipient_List property of the Notification Class object supports the address form of the Recipient component. The intent is to ensure that the IUT is able to send notifications to the specified recipient.

Test Concept: The tester shall select a single event-generating object E in the IUT that references Notification Class object N. The tester shall add an entry into the Recipient_List of the associated Notification Class object that specifies a BACnetAddress A, where A is a unicast or is a local, remote, or global broadcast address. *For devices that implement a read-only Recipient_List property for all instances of Notification Class objects and there is a Device Identifier form of the Recipient component in all instances, this test shall be skipped.*

Test Steps:

```
1.  WRITE N.RecipientList = ( {all days, all times, A, any process ID, FALSE, all transitions} )
2.  MAKE (the event generating object, E, transition)
3.  BEFORE Notification Fail Time
        RECEIVE UnconfirmedEventNotification-Request,
            DESTINATION =                       A,
            'Process Identifier' =              (the valid process ID from step 1),
            'Initiating Device Identifier' =    IUT,
            'Event Object Identifier' =         E,
            'Time Stamp' =                      (the current local time),
            'Notification Class' =              (N's instance),
            'Priority' =                        (any valid priority),
            'Event Type' =                      (any valid event type),
            'Notify Type' =                     ALARM | EVENT,
            'AckRequired' =                     TRUE | FALSE,
            'From State' =                      (any valid event state),
            'To State' =                        (any valid event state),
```

'Event Values' = (values appropriate to the event type)

[Add new test into BTL Specified Tests.]

7.3.2.21.3.X9 Read-only Recipient_List for external Notification Forwarder Objects

Purpose: This test case verifies that a read-only Notification Class object Recipient_List is configured with the content designed for external Notification Forwarder objects.

Test Concept: Read the Recipient_List of the Notification Class objects and check that the length is 1, the Recipient is local broadcast, Valid Days are all days, From Time and To Time are the entire day, Process Identifier is 0, Issue Confirmed Notification is False and Transitions is set to all transitions. This test is only applied to IUT devices that have read-only Notification Class object Recipient_List properties, and which do not contain internal Notification Forwarder objects.

Test Steps:

```
1.  READ RL = Recipient_List
2.  VERIFY (RL is a list of  length 1)
3.  VERIFY (RL.Destination = {   (1, 1, 1, 1, 1, 1, 1)--Valid Days
                                 00:00:00.0             --From Time
                                 23:59:59.99        --To Time
                                 (BACnetAddress: network-number = 0, zero length mac-address)
                                 0                  --Process Identifier
                                 False                  --Issue Confirmed Notifications
                                 (True, True, True) --Transitions
                             })
```

BTL-TP15.0-4.2.0 Tests for the claim of AE-CRL-B

Devices claiming AE-CRL-B must be subjected to this testing.

Reason for Change: Addendum 135-2012*bc* added the AE-CRL-B requirement in B-BC and B-AAC device profiles. These profiles were formerly required by the BTL in all devices with Recipient_List in Notification Class objects to support writing all forms. This document makes needed changes to update requirements for claiming the B-BC and B-AAC device profiles.

Typographic conventions: Changes in Checklist are shown by *added material in italics*~~strike-through shows removal~~

Changes to BTL Checklist:

[In BTL Checklist, extend footnote at end of Notification Class , and make one line BTL-C in clause 3.]

3.17 Notification Class

Notification Class		
	R	Base Requirements
	C[1]~~BTL-R~~	Supports DM-DDB-A
	C~~[1]~~*[2][3]*	Supports writable Recipient_List properties
	C~~[1]~~*[2]*	Supports read-only Recipient_List properties
	BTL-C[3]	*Supports AE-CRL-B*

[1] *Required if "Supports writable Recipient_List properties".*

~~[1]~~*[2]* At least one of these options must be supported.

[3] *Required if the IUT claims device profile B-AAC or B-BC.*

[In BTL Checklist, add new section at end of Alarm and Event Management as shown, after the last existing section in clause 5.]

5.X21 Alarm and Event - Configurable Recipient Lists - B

Alarm and Event - Configurable Recipient Lists - B		
	R	*Base Requirements*
	BTL-R[1]	*Supports DS-WP-B*
	R	*Supports DM-DDB-A*

[1] *The Recipient_List properties of all Notification Class and Notification Forwarder objects present in the device shall be writable.*

Changes to BTL Test Plan

[In BTL Test Plan, add new section and subsections at end of Alarm and Event Management for the AE-CRL-B BIBB]

5.X21 Alarm and Event - Configurable Recipient Lists - B

5.X21.1 Base Requirements

There are no base requirements tests for this section. Existing tests in Notification Class ensure Recipient_List in Notification Class objects support writing all forms.

5.X21.2 Supports DS-WP-B

The IUT supports DS-WP-B for its Recipient_List in Notification Class or Notification Forwarder objects.

Verify Checklist		
	Test Method	Manual
	Configuration	
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims “Supports writable Recipient_List properties” in all Notification Class objects, and if it supports Notification Forwarder objects claims “Supports writable Recipient_List properties” in all Notification Forwarder objects.
	Testing Hints	
	Notes & Results	

5.X21.3 Supports DM-DDB-A

The IUT supports DM-DDB-A.

Verify Checklist		
	Test Method	Manual
	Configuration	
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DM-DDB-A.
	Testing Hints	
	Notes & Results	

BTL-TP15.0-5.1.0 Tests for the Lighting Output object

A device including a Lighting Output object must claim Protocol_Revision 14 or higher and comply with the following section.

Overview:

Addendum 135-2010*i* added the Lighting Output object. This document makes needed changes in the BTL Test Package to claim Lighting Output object.

These changes are not contained in any SSPC proposal.

[In BTL Checklist, add Lighting Output object type to Section 3, Objects]

Support	**Listing**	**Option**
Lighting Output Object		

Support	Listing	Option
	R	Base Requirements
	R	Supports command prioritization
	R	Supports all BACnetLightingOperations
	S	Supports writable Out_Of_Service properties
	O	Supports blink-warn
	O	Supports Transition property
	O	Supports Feedback_Value property
	O	Supports Min_Actual_Value and Max_Actual_Value properties
	O	Contains an object with Reliability_Evaluation_Inhibit Property

[In BTL Test Plan, add Lighting Output object tests in section 3.X54. In the following addition of new clauses to the Test Plan, these are indicated as entirely new sections verbatim, with plain text, verbatim **bold**, or verbatim ***bold-italic*** as shown.]

3.X54 Lighting Output Object

3.X54.1 Base Requirements

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

BTL - 7.3.2.X54.21 - Lighting Output Tracking Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.2.X54.22 - Lighting Output Present Value between 0.0 and 1.0 Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X54.2 Supports Command Prioritization

135.1-2013 - 7.3.1.2 - Relinquish Default Test		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	If no object can be made to meet the configuration requirements, this test shall be skipped.

	Test Directives	
	Testing Hints	
	Notes & Results	
135.1-2013 - 7.3.1.3 - Command Prioritization Test		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X54.3 Supports all BACnetLightingOperations

BTL -7.3.2.X54.31 Lighting Command Operation NONE Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.2.X54.32 Lighting Command Operation FADE_TO Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test by using the BACnetLightingCommand without the optional fields (priority and fade-time) and check that PTY1= Lighting_Command_Default_Priority and fade-time = Default_Fade_Time
	Testing Hints	
	Notes & Results	
BTL - 7.3.2.X54.33 Lighting Command Operation RAMP_TO Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test by using the BACnetLightingCommand without the optional fields (priority and ramp-rate) and check that PTY1= Lighting_Command_Default_Priority and ramp-rate = Default_ Ramp_Rate
	Testing Hints	
	Notes & Results	
BTL - 7.3.2.X54.34 Lighting Command Operation STEP_UP Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.

	Test Directives	Repeat the test by using the BACnetLightingCommand without the optional fields (priority and ramp-rate) and check that PTY1= Lighting_Command_Default_Priority and step-increment = Default_ Step_Increment
	Testing Hints	
	Notes & Results	

BTL - 7.3.2.X54.35 Lighting Command Operation STEP_DOWN Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test by using the BACnetLightingCommand without the optional fields (priority and ramp-rate) and check that PTY1= Lighting_Command_Default_Priority and step-increment = Default_ Step_increment
	Testing Hints	
	Notes & Results	

BTL - 7.3.2.X54.36 Lighting Command Operation STEP_ON Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test by using the BACnetLightingCommand without the optional fields (priority and ramp-rate) and check that PTY1= Lighting_Command_Default_Priority and step-increment = Default_ step-increment
	Testing Hints	
	Notes & Results	

BTL - 7.3.2.X54.37 Lighting Command Operation STEP_OFF Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test by using the BACnetLightingCommand without the optional fields (priority and ramp-rate) and check that PTY1= Lighting_Command_Default_Priority and step-increment = Default_ step-increment
	Testing Hints	
	Notes & Results	

3.X54.4 Supports Writable Out_Of_Service Properties

The Out_Of_Service property in Lighting Output objects contained in the IUT are writable.

135.1-2013 - 7.3.1.1 - Out_Of_Service, Status_Flags, and Reliability Tests		
	Test Method	Manual
	Configuration	
	Test Conditionality	Must be executed.
	Test Directives	The test shall be executed using an Lighting Output object

	Testing Hints	
	Notes & Results	

3.X54.5 Supports blink-warn

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

BTL - 7.3.1.X.1 - Blink Warn WARN Command Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Must be executed using both the Present_Value and Lighting_Command commands.
	Testing Hints	
	Notes & Results	
BTL - 7.3.1.X.2 - Blink Warn WARN_OFF Command Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Must be executed using both the Present_Value and Lighting_Command commands.
	Testing Hints	
	Notes & Results	
BTL - 7.3.1.X.3 - Blink Warn WARN_RELINQUISH Command Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Must be executed using both the Present_Value and Lighting_Command commands.
	Testing Hints	
	Notes & Results	
BTL - 7.3.1.X.4 - Blink Warn STOP Command Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test with WARN_OFF and WARN_RELINQUISH commands
	Testing Hints	
	Notes & Results	
BTL - 7.3.1.X.5 - Blink Warn WARN Command Failure Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.

	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.1.X.6 - Blink Warn WARN_OFF Command Failure Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.1.X.7 - Blink Warn WARN_RELINQUISH Command Failure Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.1.X.8 - Blink Warn WARN_OFF Command Halted Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.1.X.9 - Blink Warn WARN_RELINQUISH Command Halted Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X54.6 Supports Transition property

The supports transition section and tests for: RAMP transition executes a ramp operation from the Tracking_Value to the target level using the ramp rate specified in Default_Ramp_Rate.

BTL - 7.3.2.X54.41 Transition None Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

	Notes & Results	
BTL - 7.3.2.X54.42 Transition Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X54.7 Supports Feedback_Value property

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

BTL - 7.3.2.X54.51 - Feedback_Value Clamping Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X54.8 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.

BTL - 7.3.2.X54.61 - Min_Actual_Value and Max_Actual_Value Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.2.X54.62 - Min_Actual_Value and Max_Actual_Value ScalingTest		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X54.9 Contains an object with Reliability_Evaluation_Inhibit Property

The IUT contains, or can be made to contain, a Reliability_Evaluation_Inhibit property that is configurable to a value of TRUE.

BTL - 7.3.1.X8.1 - Reliability_Evaluation_Inhibit Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If no object exists in the IUT for which fault conditions can be generated then this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.1.X8.2 - Reliability_Evaluation_Inhibit Summarization Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If no object exists in the IUT for which fault conditions can be generated then this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	

Changes:

[In BTL Specified Tests, add new Lighting Output object specific tests in section 7.3.2.X54]

7.3.2.X54.21 - Lighting Output Tracking Test

Purpose: To verify that the Tracking_Value property follows the Present_Value property.

Test Concept: Write to the Present_Value of a Lighting Output object, O1, and verify that the Tracking_Value property follows Present_Value once In-Progress returns to IDLE.

Configuration Requirements: The IUT shall be configured with a lighting output O1 that can be observed during the test. 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 and Out_Of_Service = FALSE. Any scaling information that may be needed to verify that the value is reasonable shall also be provided.

Test Steps:

```
1.    WRITE Present_Value = 100, PRIORITY = PTY1
2.    VERIFY Present_Value = 100
3.    WHILE (In_Progress <> IDLE) DO {
      }
4.    VERIFY Tracking_Value = 100
5.    WRITE   Present_Value = 1, PRIORITY = PTY1
6.    VERIFY Present_Value = 1
7.    WHILE (In_Progress <> IDLE) DO {
      }
8.    VERIFY Tracking_Value = 1
9.    WRITE Present_Value = 0, PRIORITY = PTY1
10.   VERIFY Present_Value = 0
11.   WHILE (In_Progress <> IDLE) DO {
```

}

12. VERIFY Tracking_Value = 0

7.3.2.X54.22 - Lighting Output Present Value between 0.0 and 1.0 Test

Purpose: To verify that writing a value numerically greater than 0.0 but less than 1.0 to Present_Value shall result in Present_Value taking on the value 1.0.

Test Concept: Select a value, V1, which is numerically greater than 0.0 and less than 1.0. Write V1 to Present_Value and verify that Present_Value takes on the value 1.0.

Configuration Requirements: The Lighting Output object, 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. Present_Value shall be different from 1.0.

Test Steps:

1. VERIFY Present_Value <> 1.0
2. WRITE Present_Value = a value numerically greater than 0.0 but less than 1.0
3. VERIFY Present_Value = 1.0

7.3.2.X54.31 Lighting Command Operation NONE Test

Purpose: To verify that the IUT can execute WriteProperty service requests when an attempt is made to write a value that is outside of the supported range.

Test Concept: The TD writes the Lighting Command Operation NONE to the IUT, and expects Error Class of PROPERTY and an Error Code of VALUE_OUT_OF_RANGE

Test Steps:

1. VERIFY (Object1), P1 = (the value defined for this property in the EPICS)~~,~~
2. TRANSMIT WriteProperty-Request,
    'Object Identifier' = O1
    'Property Identifier' = Lighting_Command
    'Property Value' = NONE
3. RECEIVE BACnet-Error PDU,
    Error Class = PROPERTY,
    Error Code = VALUE_OUT_OF_RANGE
4. VERIFY (Object1), Lighting_Command = (the value defined for this property in the EPICS)

7.3.2.X54.32 Lighting Command Operation FADE_TO Test

Purpose: To verify the correct operation of FADE_TO lighting command by observing the value of Present_Value, In_Progress and Tracking_Value.

Test Concept: The TD writes to the Present_Value at each end of the range (i.e. 0% or 100%), and then writes to the Lighting Command Operation with FADE_TO with a long enough fade-time to allow In_Progress and Tracking_Value to be observed while set to FADE_ACTIVE. The Tracking_Value will be checked at the end of the fade to verify that it tracked the target level. The IUT shall be tested for fade up (0% to 100%) and fade down (100% to 0%).

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. V1 > 1 and V2 < 100%

Test Steps:

-- Start with 0% Present_Value to test fade up

1. WRITE Present_Value = 0, ARRAY_INDEX = PTY1

2. VERIFY Present_Value = 0
3. WAIT **Internal Processing Fail Time**
4. VERIFY Tracking_Value = 0

-- Write a FADE_TO command (operation, target-level, priority, fade-time)
5. WRITE Lighting_Command = (FADE_TO, V1, PTY1, FT)
6. WAIT **Internal Processing Fail Time**
7. VERIFY Priority_ Array = V1, ARRAY_INDEX = PTY1
8. VERIFY Present_Value = V1

-- In a half way of fading up, check In_Progress and Tracking_Value
9. WAIT FT/2
10. VERIFY In_Progress = FADE_ACTIVE,
11. VERIFY Tracking_Value ~=V1 / 2
12. WAIT FT/2

-- When fading up is completed, check In_Progress and Tracking_Value
13. VERIFY In_Progress = IDLE
14. VERIFY Tracking_Value = V1

-- Now repeat the test with 100% Present_Value to test fade down
15. WRITE Present_Value = 100, ARRAY_INDEX = PTY1
16. VERIFY Present_Value = 100
17. WAIT **Internal Processing Fail Time**
18. VERIFY Tracking_Value = 100

-- Write a FADE_TO command (operation, target-level, priority, fade-time)
19. WRITE Lighting_Command = (FADE_TO, V2, PTY1, FT)
20. WAIT **Internal Processing Fail Time**
21. VERIFY Priority_ Array = V2, ARRAY_INDEX = PTY1
22. VERIFY Present_Value = V2

-- In a half way of fading down, check In_Progress and Tracking_Value
23. WAIT FT/2
24. VERIFY In_Progress = FADE_ACTIVE,
25. VERIFY Tracking_Value ~=V1 / 2
26. WAIT FT/2

-- When fading down is completed, check In_Progress and Tracking_Value
27. VERIFY In_Progress = IDLE
28. VERIFY Tracking_Value = V2

7.3.2.X54.33 Lighting Command Operation RAMP_TO Test

Purpose: To verify the correct operation of RAMP_TO lighting command by observing the value of Present_Value, In_Progress and Tracking_Value.

Test Concept: The TD writes to Present_Value at each end of the range (i.e. 0% or 100%), and then writes to the Lighting Command Operation with RAMP_TO with a slow enough ramp rate to allow In_Progress and Tracking_Value to be observed while set to RAMP_ACTIVE. The Tracking_Value will be checked at the end of the ramp to verify that it tracked the target level. The IUT shall be tested for ramp up (0% to 100%) and ramp down (100% to 0%).

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. V1 > 1 and V2 < 100%

Test Steps:

```
-- Start with 0% Present_Value to test ramp up
1.  WRITE Present_Value = 0, ARRAY_INDEX = PTY1
2.  VERIFY Present_Value = 0
3.  WAIT Internal Processing Fail Time
4.  VERIFY Tracking_Value = 0

-- Write a RAMP_TO command (operation, target-value, priority, ramp-rate)
5.  WRITE Lighting_Command = (RAMP_TO, V1, PTY1, any valid rate)
6.  WAIT Internal Processing Fail Time
7.  VERIFY Priority_ Array = V1, ARRAY_INDEX = PTY1
8.  VERIFY Present_Value =V1

-- Check In_Progress while ramping up
9.  VERIFY In_Progress = RAMP_ACTIVE

-- Make sure that Tracking_Value increases with the ramp-rate
10. WHILE (In_Progress <> IDLE) DO {
11. VERIFY Tracking_Value > 0 < V1
12. CHECK (Tracking_Value is increasing with the ramp-rate)}

-- When ramping up is completed, check In_Progress and Tracking_Value
13. VERIFY In_Progress = IDLE
14. VERIFY Tracking_Value = V1

--  Now repeat the test with 100% Present_Value to test ramp down
15. WRITE Present_Value = 100, ARRAY_INDEX = PTY1
16. VERIFY Present_Value = 100
17. WAIT Internal Processing Fail Time
18. VERIFY Tracking_Value = 100

-- Write a RAMP_TO command (operation, target-value, priority, ramp-rate)
19. WRITE Lighting_Command = (RAMP_TO, V2, PTY1, any valid rate)
20. WAIT Internal Processing Fail Time
21. VERIFY Priority_ Array = V2, ARRAY_INDEX = PTY1
22. VERIFY Present_Value = V2

-- Check In_Progress while ramping up
23. VERIFY In_Progress = RAMP_ACTIVE,

-- Make sure that Tracking_Value decreases with the ramp-rate
24. WHILE (In_Progress <> RAMP_ACTIVE) DO {
25. VERIFY Tracking_Value < 0 > V2
26. CHECK (Tracking_Value is decreasing with the ramp-rate)}

-- Check In_Progress and Tracking_Value
27. VERIFY In_Progress = IDLE
28. VERIFY Tracking_Value = V2
```

7.3.2.X54.34 Lighting Command Operation STEP_UP Test

Purpose: To verify the correct operation of STEP_UP lighting command by observing the value of Present_Value, In_Progress and Tracking_Value.

Test Concept: The TD writes to Present_Value at 0%, and then writes to the Lighting Command Operation with STEP_UP and any step increment. The Tracking_Value shall remain at 0% to ignore the operation. Next, the TD writes to Present_Value at 1%, and then writes to the Lighting Command Operation with STEP_UP and a step increment greater than 99%, the Tracking_Value shall be 100%. The TD writes to Present_Value at 1%, and then writes to the Lighting Command Operation with STEP_UP and a step increment less than 99%, the Tracking_Value shall be 1% plus the step increment.

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.

Test Steps:

```
-- Start with 0% Present_Value
1.  WRITE Present_Value = 0, ARRAY_INDEX = PTY1
2.  VERIFY Present_Value = 0
3.  WAIT **Internal Processing Fail Time**
4.  VERIFY Tracking_Value = 0

-- Write a STEP_UP command (operation, priority, step-increment)
5.  WRITE Lighting_Command = (STEP_UP, PTY1, any valid value)
6.  WAIT **Internal Processing Fail Time**

-- Confirm that the command was ignored since Tracking_Value was 0
7.  VERIFY Priority_ Array = 0, ARRAY_INDEX = PTY1
8.  VERIFY Present_Value = 0
9.  VERIFY Tracking_Value = 0

-- Now test with Tracking_Value >0
10. WRITE Present_Value = 1, ARRAY_INDEX = PTY1
11. VERIFY Present_Value = 1
12. WAIT **Internal Processing Fail Time**
13. VERIFY Tracking_Value = 1

-- Keep stepping up while continuously checking Priority_Array, Present_Value and Tracking_Value
14. REPEAT X = (1 through (100 - step-increment) by step-increment) DO{
        WRITE Lighting_Command = (STEP_UP, PTY1, any valid value)
        WAIT **Internal Processing Fail Time**
        VERIFY Priority_ Array = X + step-increment, ARRAY_INDEX = PTY1
        VERIFY Present_Value = X + step-increment
        VERIFY Tracking_Value = X + step-increment

-- Now step up one more time to confirm that the values will not exceed 100
15. WRITE Lighting_Command = (STEP_UP, PTY1, any valid value)
16. WAIT **Internal Processing Fail Time**
17. VERIFY Priority_ Array = 100, ARRAY_INDEX = PTY1
18. VERIFY Present_Value =100
19. VERIFY Tracking_Value = 100
```

7.3.2.X54.35 Lighting Command Operation STEP_ DOWN Test

Purpose: To verify that writing this Lighting Command Operation is reflected in the Tracking_Value, that writes resulting in a step below 1% are limited to 1%, and that this command is ignored if the Tracking_Value is 0.0%.

Test Concept: The TD writes to Present_Value at 0%, and then writes to the Lighting Command Operation with STEP_DOWN and any step increment. The Tracking_Value shall remain at 0%. The TD writes to Present_Value at

100%, and then writes to the Lighting Command Operation with STEP_DOWN and a step increment greater than 99%, the Tracking_Value shall be 1%. The TD writes to Present_Value at 100%, and then writes to the Lighting Command Operation with STEP_DOWN and a step increment less than 99%, the Tracking_Value shall be 100% minus the step increment.

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.

Test Steps:
-- Start with 0% Present_Value
1. WRITE Present_Value = 0, ARRAY_INDEX = PTY1
2. VERIFY Present_Value = 0
3. WAIT **Internal Processing Fail Time**
4. VERIFY Tracking_Value = 0

-- Write a STEP_DOWN command (operation, priority, step-increment)
5. WRITE Lighting_Command = (STEP_ DOWN, PTY1, any valid value)
6. WAIT **Internal Processing Fail Time**

-- Confirm that the command was ignored since Tracking_Value was 0
7. VERIFY Priority_ Array = 0, ARRAY_INDEX = PTY1
8. VERIFY Present_Value = 0
9. VERIFY Tracking_Value = 0

-- Now test with Tracking_Value = 100
10. WRITE Present_Value = 100, ARRAY_INDEX = PTY1
11. VERIFY Present_Value = 100
12. WAIT **Internal Processing Fail Time**
13. VERIFY Tracking_Value =100

-- Keep stepping down while continuously checking Priority_Array, Present_Value and Tracking_Value

```
14. REPEAT X = (100 through (1 + step-increment) by step-increment) DO{
        WRITE Lighting_Command = (STEP_ DOWN, PTY1, any valid value)
        WAIT Internal Processing Fail Time
        VERIFY Priority_ Array = X - step-increment, ARRAY_INDEX = PTY1
        VERIFY Present_Value = X - step-increment
        VERIFY Tracking_Value = X - step-increment
```

-- Now step down one more time to confirm that the values will not go down below 1
15. WRITE Lighting_Command = (STEP_ DOWN, PTY1, any valid value)
16. WAIT **Internal Processing Fail Time**
17. VERIFY Priority_ Array = 1, ARRAY_INDEX = PTY1
18. VERIFY Present_Value =1
19. VERIFY Tracking_Value = 1

7.3.2.X54.36 Lighting Command Operation STEP_ON Test

Purpose: To verify that writing this Lighting Command Operation is reflected in the Tracking_Value, that this command will set the Tracking_Value to 1% if the Tracking_Value is 0.0%, and that it otherwise adheres to STEP_UP.

Test Concept: The TD writes to Present_Value at 0%, and then writes to the Lighting Command Operation with STEP_UP and any step increment. The Tracking_Value shall be 1%. The TD writes to Present_Value at 1%, and then writes to the Lighting Command Operation with STEP_UP and a step increment greater than 99%, the Tracking_Value shall be 100%. The TD writes to Present_Value at 1%, and then writes to the Lighting Command

Operation with STEP_UP and a step increment less than 99%, the Tracking_Value shall be 1% plus the step increment.

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.

Test Steps:

```
-- Start with 0% Present_Value
1.  WRITE Present_Value = 0, ARRAY_INDEX = PTY1
2.  VERIFY Present_Value = 0
3.  WAIT Internal Processing Fail Time
4.  VERIFY Tracking_Value = 0

-- Write a STEP_ON command (operation, priority, step-increment)
5.  WRITE Lighting_Command = (STEP_ON, PTY1, any valid values)
6.  WAIT Internal Processing Fail Time

-- Confirm that the Present_Value and Tracking_Value became 1
7.  VERIFY Priority_ Array = 1, ARRAY_INDEX = PTY1
8.  VERIFY Present_Value = 1
9.  VERIFY Tracking_Value = 1

-- Keep stepping on while continuously checking Priority_Array, Present_Value and Tracking_Value
10. REPEAT X = (1 through (100 – step-increment))
            WRITE Lighting_Command = (STEP_ON, PTY1, any valid values)
            WAIT Internal Processing Fail Time
            VERIFY Priority_ Array = X + step-increment, ARRAY_INDEX = PTY1
            VERIFY Present_Value = X + step-increment
            VERIFY Tracking_Value = X + step-increment

-- Now step on one more time to confirm that the values will not exceed 100
11. WRITE Lighting_Command = (STEP_ON, PTY1, any valid values)
12. WAIT Internal Processing Fail Time
13. VERIFY Priority_ Array = 100, ARRAY_INDEX = PTY1
14. VERIFY Present_Value =100
15. VERIFY Tracking_Value = 100
```

7.3.2.X54.37 Lighting Command Operation STEP_ OFF Test

Purpose: To verify that writing this Lighting Command Operation is reflected in the Tracking_Value, that writes resulting in a step below 1% are limited to 1%, and that this command is ignored if the Tracking_Value is 0.0%.

Test Concept: The TD writes to Present_Value at 0%, and then writes to the Lighting Command Operation with STEP_DOWN and any step increment. The Tracking_Value shall remain at 0%. The TD writes to Present_Value at 100%, and then writes to the Lighting Command Operation with STEP_DOWN and a step increment greater than 99%, the Tracking_Value shall be 1%. The TD writes to Present_Value at 100%, and then writes to the Lighting Command Operation with STEP_DOWN and a step increment less than 99%, the Tracking_Value shall be 100% minus the step increment.

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.

Test Steps:

-- Start with 0% Present_Value
1. WRITE Present_Value = 0, ARRAY_INDEX = PTY1
2. VERIFY Present_Value = 0
3. WAIT **Internal Processing Fail Time**
4. VERIFY Tracking_Value = 0

-- Write a STEP_OFF command (operation, priority, step-increment)
5. WRITE Lighting_Command = (STEP_ OFF, PTY1, step-increment)
6. WAIT **Internal Processing Fail Time**

-- Confirm that the command was ignored since Tracking_Value was 0
7. VERIFY Priority_ Array = 0, ARRAY_INDEX = PTY1
8. VERIFY Present_Value = 0
9. VERIFY Tracking_Value = 0

-- Now test with Tracking_Value = 100
10. WRITE Present_Value = 100, ARRAY_INDEX = PTY1
11. VERIFY Present_Value = 100
12. WAIT **Internal Processing Fail Time**
13. VERIFY Tracking_Value =100

-- Keep stepping off while continuously checking Priority_Array, Present_Value and Tracking_Value
14. REPEAT X = (100 through (1 + step-increment))
    - WRITE Lighting_Command = (STEP_ OFF, PTY1, step-increment)
    - WAIT **Internal Processing Fail Time**
    - VERIFY Priority_ Array = X - step-increment, ARRAY_INDEX = PTY1
    - VERIFY Present_Value = X - step-increment
    - VERIFY Tracking_Value = X - step-increment

-- Confirm that the Present_Value and Tracking_Value become 0 when STEP OFF command is executed while Tracking_Value is 1

15. WRITE Lighting_Command = (STEP_ OFF, PTY1, step-increment)
16. WAIT **Internal Processing Fail Time**
17. VERIFY Priority_ Array = 0, ARRAY_INDEX = PTY1
18. VERIFY Present_Value = 0
19. VERIFY Tracking_Value = 0

7.3.2.X54.41 Transition None test

Purpose: To verify that the Tracking_Value property immediately follows the Present_Value property if Transition is NONE.

Test Concept: Setup a Lighting Output object, O1, to use its complete supported value range. Set Present_Value to the highest supported value, and then to the lowest supported value, verifying that there is no delay in the transitions.

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. If present, Min_Actual_Value shall be set to 1, and Max_Actual_Value shall be set to 100. Transition shall be set to NONE.

Test Steps:
1. VERIFY Transition = NONE
2. VERIFY In_Progress = IDLE
3. WRITE Present_Value = 100, ARRAY_INDEX = PTY1
4. VERIFY In_Progress = IDLE

5. VERIFY Tracking_Value = 100
6. WRITE Present_Value = 1, ARRAY_INDEX = PTY1
7. VERIFY In_Progress = IDLE
8. VERIFY Tracking_Value = 1

7.3.2.X54.42 Transition Test

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 )
10. WAIT W1
11. VERIFY In_Progress = IDLE
12. VERIFY Tracking_Value = V1
13. WRITE Present_Value = V2, ARRAY_INDEX = PTY1
14. VERIFY Present_Value = V2
15. WAIT W2
16. VERIFY Tracking_Value ~= (V2 + V1) / 2
17. VERIFY In_Progress = A
18.     CHECK (the physical output is fading V1 to V2 )
19.     WAIT W2
20. VERIFY In_Progress = IDLE
21. VERIFY Tracking_Value = V2
```

7.3.2.X54.51 - Feedback_Value Clamping Test

Purpose: To verify that the Feedback_Value remains in the normalized range when the physical lighting output is outside the normalized range.

Test Concept: Set the normalized range to be the largest range supported by the device. Make the physical output be above the normalized range by setting it to the maximum supported value and then shrinking the normalized range. The Feedback_Value is immediately tested to verify that it takes on the value 100.

Reset the normalized range. Make the physical output be below the normalized range by setting it to the minimum supported value and then shrinking the normalized range. The Feedback_Value is immediately tested to verify that it takes on the value 1.

Configuration Requirements: The Lighting Output object, O1, shall be configured to transition slowly when Present_Value changes, such as by ramping, fading or stepping, if possible.

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.

Test Steps:

```
-- Verify Feedback_Value when output is above Max_Actual_Value
1.    WRITE Max_Actual_Value = 100
2.    WRITE Min_Actual_Value = 1
3.    WRITE Present_Value = 100, PRIORITY = PTY1
4.    WHILE In_Progress <> IDLE {}
5.    WRITE Max_Actual_Value = (Lowest supported Max_Actual_Value)
6.    VERIFY Feedback_Value = 100

-- Verify Feedback_Value when output is below Min_Actual_Value
7.    WRITE Max_Actual_Value = 100
8.    WRITE Min_Actual_Value = 1
9.    WRITE Present_Value = 1, PRIORITY = PTY1
10.   WHILE In_Progress <> IDLE {}
11.   WRITE Min_Actual_Value = (Highest supported Min_Actual_Value)
12.   VERIFY Feedback_Value = 1
```

7.3.2.X54.61 Min_Actual_Value and Max_Actual_Value Test

Purpose: To verify that Min_Actual_Value remains less than Max_Actual_Value and within the allowable range when either is written to a value that would violate these conditions.

Test Concept: Write a value to Min_Actual_Value which is larger than Max_Actual_Value. Verify that Max_Actual_Value became equal to Min_Actual_Value. Next, write a value to Max_Actual_Value which is less than Min_Actual_Value. Verify that Min_Actual_Value became equal to Max_Actual_Value.

Verify that neither Min_Actual_Value nor Max_Actual_Value will accept a value outside the range 1.0 to 100.0.

Configuration Requirements: The IUT shall be configured with a lighting output, O1. Min_Actual_Value shall be set to a value less than Max_Actual_Value, and Max_Actual_Value shall be within the allowable range for Min_Actual_Value and not equal to Min_Actual_Value's maximum supported value. If the IUT cannot be configured to meet these requirements, then this test shall be skipped.

Test Steps:

```
1.    V1 = READ Max_Actual_Value
2.    WRITE Min_Actual_Value = V2, a value greater than V1
3.    VERIFY Max_Actual_Value = V2
4.    WRITE Max_Actual_Value = V3, a value less than V2
5.    VERIFY Min_Actual_Value = V3
6.    TRANSMIT WritePropertyRequest
          'Object Identifier' =  O1,
```

‘Property Identifier’ = Min_Actual_Value,
‘Property Value’ = (any value outside the range 1.0 to 100.0)

7. RECEIVE BACnet-Error-PDU,
   Error Class = PROPERTY,
   Error Code = VALUE_OUT_OF_RANGE
8. TRANSMIT WritePropertyRequest
   'Object Identifier' = O1,
   ‘Property Identifier’ = Max_Actual_Value,
   ‘Property Value’ = (any value outside the range 1.0 to 100.0)

9.. RECEIVE BACnet-Error-PDU,
   Error Class = PROPERTY,
   Error Code = VALUE_OUT_OF_RANGE

7.3.2.X54.62 Min_Actual_Value and Max_Actual_Value Scaling Test

Purpose: To verify that the physical output level changes to the expected scaled value as Present_Value changes.

Test Concept: Set Min_Actual_Value to a value other than the lowest supported minimum value, and set Max_Actual_Value to a value other than the highest support value but larger than Min_Actual_Value.

Then write 1.0 to Present_Value and measure the physical output. Repeat the procedure to measure the physical output after writing 100.0 to Present_Value. After obtaining these upper and lower bound values, write a value between 1.0 and 100.0, measure the physical output, and confirm that the measured value is approximately the same as the expected scaled value.

Configuration Requirements: The IUT shall be configured with a lighting output, O1 that can be observed during the test. 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 and Out_Of_Service = FALSE.

Test Steps:

1. WRITE Min_Actual_Value = (a supported value that is not the lowest supported value)
2. WRITE Max_Actual_Value = (a supported value which is not the highest support value)
3. WRITE Present_Value = 1.0, ARRAY_INDEX = PTY1
4. CHECK(the value of the physical output is Min_Actual_Value)
5. WRITE Present_Value = 100.0, ARRAY_INDEX = PTY1
6. CHECK(the value of the physical output is Max_Actual_Value)
7. WRITE Present_Value = (V1, a value between 1.0 and 100.0 exclusive), ARRAY_INDEX = PTY1
8. MAKE(measure the value of the physical output and record in MV)
9. CHECK (MV ~= Min_Actual_Value + (V1 / 100) * (Max_Actual_Value – Min_Actual_Value))

[In Interim_Tests_15, there are other referenced non-object specific tests for Blink in section 7.3.1.X, applicable to both Lighting Output or Binary Lighting Output objects, were added in BTL-15.0-2.1.0]

BTL-TP15.1-5.2.0 Lighting Output object for DS-COV-B

[In BTL Interim_Tests_15.1, add the below DM-COV-B Test Plan items]

Changes:

[In BTL Test Plan, add “Supports COV for Lighting Output Objects “ tests” in section 4.10.X. In the following modification of clauses of the Test Plan, further changes in a test name, which had already been earlier changed for the version in BTL Specified Tests (in ***bold-italic*** as shown), are indicated in *red-italic*, with the rest of these new clauses of the Test Plan in plain text, verbatim **bold**, or verbatim ***bold-italic*** as shown.]

4.10.X54 Supports COV for Lighting Output Objects

The IUT supports change of value notifications for at least one object of type Lighting Output.

BTL - 8.2.1 - Change of Value Notification from an Analog Object Present_Value Property		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	This may be skipped if 8.3.1 is executed against a Lighting Output object.
	Notes & Results	
BTL - 8.2.2 - Change of Value Notification from an Analog Object Status_Flags Property		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***. The selected object must be a Lighting Output.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	This may be skipped if 8.3.2 is executed against a Lighting Output object.
	Notes & Results	
BTL - 8.3.1 - Change of Value Notification from an Analog Object Present_Value Property		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***. The selected object must be a Lighting Output.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	This may be skipped if 8.2.1 is executed against a Lighting Output object.
	Notes & Results	
BTL - 8.3.2 - Change of Value Notification from an Analog Object Status_Flags Property		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***. The selected object must be a Lighting Output.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	This may be skipped if 8.2.2 is executed against a Lighting Output

		object.
	Notes & Results	

Changes:

[In BTL Specified Tests, modify the test 8.2.1, 8.2.2, 8.3.1, 8.3.2 to test against Lighting Output]

8.2.1 Change of Value Notification from an Analog Object Present_Value Property

Reason for Change: Add more primitive value objects. Updated description of the 'List of Values' to improve readability. Updated 'Configuration Requirements'.

Purpose: To verify that the IUT can initiate ConfirmedCOVNotification service requests conveying a change of the Present_Value property of Analog Input, Analog Output, *Lighting Output*,~~and~~ Analog Value, *Large Analog Value, Integer Value, and Positive Integer Value* objects.

Test Concept: A subscription for COV notifications is established, using a Lifetime of L. L shall be set to a value less than 24 hours and large enough to complete the test. The Present_Value of the monitored object is changed by an amount less than the COV increment and it is verified that no COV notification is received. The Present_Value is then changed by an amount greater than the COV increment and a notification shall be received. The Present_Value may be changed using the WriteProperty service or by another means such as changing the input signal represented by an Analog Input object. For some implementations it may be necessary to write to the Out_Of_Service property first to accomplish this task. For implementations where it is not possible to write to these properties at all the vendor shall provide an alternative trigger mechanism to accomplish this task. All of these methods are equally acceptable.

Configuration Requirements: At the beginning of the test, the Out_Of_Service property shall have a value of FALSE. *Select an object where Present_Value is not expected to change outside the tester's control by more than COV_Increment or which has a writable Out_Of_Service.*

Test Steps:

REPEAT X = (one supported object of each type from the set Analog Input, Analog Output, ~~and~~ Analog Value, *Large Analog Value, Integer Value, and Positive Integer Value*) DO {

1. TRANSMIT SubscribeCOV-Request,
    'Subscriber Process Identifier' = (any value > 0 chosen by the TD),
    'Monitored Object Identifier' = X,
    'Issue Confirmed Notifications' = TRUE,
    'Lifetime' = L
2. RECEIVE BACnet-SimpleACK-PDU
3. BEFORE **Notification Fail Time**
    RECEIVE ConfirmedCOVNotification-Request,
    'Subscriber Process Identifier' = (the same value used in step 1),
    'Initiating Device Identifier' = IUT,
    'Monitored Object Identifier' = X,
    'Time Remaining' = (any value appropriate for the Lifetime selected),
    'List of Values' = (the initial Present_Value and initial Status_Flags)
4. TRANSMIT BACnet-SimpleACK-PDU
5. TRANSMIT ReadProperty-Request,
    'Object Identifier' = X,
    'Property Identifier' = COV_Increment
6. RECEIVE BACnet-ComplexACK-PDU,
    'Object Identifier' = X,
    'Property Identifier' = COV_Increment,

```
        'Property Value' =                    (a value "increment" that will be used below)
7.  IF (Out_Of_Service is writable) THEN
        WRITE X, Out_Of_Service = TRUE

        BEFORE Notification Fail Time
            RECEIVE ConfirmedCOVNotification-Request,
                'Subscriber Process Identifier' =   (the same value used in step 1),
                'Initiating Device Identifier' =    IUT,
                'Monitored Object Identifier' =     X,
                'Time Remaining' =                  (any value appropriate for the Lifetime selected),
                'List of Values' =                  (ReportedPV =~~any value appropriate for~~ the current Present_Value, ~~and~~ new Status_Flags)
        TRANSMIT BACnet-SimpleACK-PDU
8.  IF (Present_Value is now writable) THEN
        WRITE X, Present_Value = (any value that differs ~~from "initial Present_Value"~~ ReportedPV by less than "increment")
            ELSE
        MAKE (Present_Value  = any value that differs from ~~"initial Present_Value"~~ ReportedPV by less than "increment")
9.  WAIT Notification Fail Time
10. CHECK (verify that no COV notification was transmitted)
11. IF (Present_Value is now writable) THEN
        WRITE X, Present_Value = (any value that differs from ~~"initial Present_Value"~~ ReportedPV by an amount greater than "increment")
~~        RECEIVE BACnet-SimpleACK-PDU~~
    ELSE
        MAKE (Present_Value  = any value that differs from ~~"initial Present_Value"~~ ReportedPV by an amount greater than "increment")
12. BEFORE NotificationFailTime
        RECEIVE ConfirmedCOVNotification-Request,
            'Subscriber Process Identifier' =       (the same value used in step 1),
            'Initiating Device Identifier' =        IUT,
            'Monitored Object Identifier' =         X,
            'Time Remaining' =                      (any value appropriate for the Lifetime selected),
            'List of Values' =                      (the new Present_Value and new Status_Flags)
13. TRANSMIT BACnet-SimpleACK-PDU
14. TRANSMIT SubscribeCOV-Request,
        'Subscriber Process Identifier' =           (the same value used in step 1),
        'Monitored Object Identifier' =             X
15. RECEIVE BACnet-SimpleACK-PDU
16. IF (Out_Of_Service is writable) THEN
        WRITE X, Out_Of_Service =                   FALSE
~~        RECEIVE BACnet-SimpleACK-PDU~~
```

8.2.2 Change of Value Notification from an Analog Object Status_Flags Property

Reason for Change: Add more primitive value objects. Updated 'Configuration Requirements'. Removed extraneous SimpleACKs after WRITE statements. Updated descriptive text for 'List of Value' property.

Purpose: To verify that the IUT can initiate ConfirmedCOVNotification service requests conveying a change of the Status_Flags property of Analog Input, Analog Output, *Lighting Output*,~~and~~ Analog Value, *Large Analog Value, Integer Value, and Positive Integer Value* objects.

Test Concept: A subscription for COV notifications is established, using a Lifetime of L. L shall be set to a value less than 24 hours and large enough to complete the test. The Status_Flags property of the monitored object is then changed and a notification shall be received. The value of the Status-Flags property can be changed by using the

WriteProperty service or by another means. For some implementations writing to the Out_Of_Service property will accomplish this task. For implementations where it is not possible to write to Status_Flags or Out_Of_Service or change the Status_Flags by any other means, this test shall be skipped

Configuration Requirements: At the beginning of the test, the Out_Of_Service property shall have a value of FALSE. *Select an object where Present_Value is not expected to change outside the tester's control by more than COV_Increment or which has a writable Out_Of_Service.*

Test Steps:

REPEAT X = (one supported object of each type from the set Analog Input, Analog Output, ~~and~~ Analog Value, *Lighting Output*, *Large Analog Value*, *Integer Value*, *and Positive Integer Value*) DO {

1. TRANSMIT SubscribeCOV-Request,
    'Subscriber Process Identifier' = (any value > 0 chosen by the TD),
    'Monitored Object Identifier' = X,
    'Issue Confirmed Notifications' = TRUE,
    'Lifetime' = L
2. RECEIVE BACnet-SimpleACK-PDU
3. BEFORE **Notification Fail Time**
    RECEIVE ConfirmedCOVNotification-Request,
    'Subscriber Process Identifier' = (the same value used in step 1),
    'Initiating Device Identifier' = IUT,
    'Monitored Object Identifier' = X,
    'Time Remaining' = (any value appropriate for the Lifetime selected),
    'List of Values' = (the initial Present_Value and initial Status_Flags)
4. TRANSMIT BACnet-SimpleACK-PDU

5. WRITE X, Out_Of_Service = TRUE | WRITE X, Status_Flags = (a value that differs from initial Status_Flags) |
    MAKE (Status_Flags = any value that differs from initial Status_Flags)
~~2. IF (WriteProperty is used in step 5) THEN~~
    ~~RECEIVE BACnet-SimpleACK-PDU~~
7. BEFORE **Notification Fail Time**
    RECEIVE ConfirmedCOVNotification-Request,
        'Subscriber Process Identifier' = (the same value used in step 1),
        'Initiating Device Identifier' = IUT,
        'Monitored Object Identifier' = X,
        'Time Remaining' = (any value appropriate for the Lifetime selected),
        'List of Values' = (~~the initial~~*the current* Present_Value and new Status_Flags)
8. TRANSMIT BACnet-SimpleACK-PDU
9. TRANSMIT SubscribeCOV-Request,
    'Subscriber Process Identifier' = (the same value used in step 1),
    'Monitored Object Identifier' = X
10. RECEIVE BACnet-SimpleACK-PDU
11. IF (Out_Of_Service was changed in step 5) THEN
    WRITE X, Out_Of_Service = FALSE
    ~~RECEIVE BACnet-SimpleACK-PDU~~

8.3.1 Change of Value Notification from an Analog Object Present_Value Property

Purpose: To verify that the IUT can initiate UnconfirmedCOVNotification service requests conveying a change of the Present_Value property of Analog Input, Analog Output, ~~and~~ Analog Value, *Large Analog Value*, *Integer Value*, ~~*and*~~ *Positive Integer Value*, *and Lighting Output* objects.

Test Steps: The steps for this test case are identical to the test steps in 8.2.3 except that the SubscribeCOV service request in step 1 shall have a value of FALSE for the 'Issue Confirmed Notifications' parameter, all of the ConfirmedCOVNotification requests shall be UnconfirmedCOVNotification requests, and there is no acknowledgment of the unconfirmed services. The MAC address used for the notification message shall be such that the TD is one of the recipients.

8.3.2 Change of Value Notification from an Analog Object Status_Flags Property

Purpose: To verify that the IUT can initiate UnconfirmedCOVNotification service requests conveying a change of the Status_Flags property of Analog Input, Analog Output, ~~and~~ Analog Value, *Large Analog Value*, *Integer Value*, ~~*and*~~ *Positive Integer Value, and Lighting Output* objects.

Test Steps: The steps for this test case are identical to the test steps in 8.2.4 except that the SubscribeCOV service request in step 1 shall have a value of FALSE for the 'Issue Confirmed Notifications' parameter, all of the ConfirmedCOVNotification requests shall be UnconfirmedCOVNotification requests, and there is no acknowledgment of the unconfirmed services. The MAC address used for the notification message shall be such that the TD is one of the recipients.

BTL-TP15.0-6.1.0 Tests for the claim of DS-COVP-B

A device claiming DS-COVP-B at Protocol_Revision 2 or higher shall comply with the following section.

Overview:

Addendum 135-1995*c* added the SubscribeCOVProperty service. This document makes needed changes in the BTL Test Package to claim the DS-COVP-B BIBB.

These changes adapt and extend some existing tests defined in 135.1.

[In Checklist, add DS-COVP-B just after existing DS-COVP-A]

4 Data Sharing

Data Sharing - Change Of Value Property – B		
	R	Base Requirements
	R	Supports COVP Lifetimes up to 8 hours in duration
	R	Supports COVP for Status_Flags changes
	C[1]	Supports COVP for non-array property
	C[1]	Supports COVP for array element
	C[1]	Supports COVP for the size of an array
	C[1]	Supports COVP for whole array
	O	Supports COVP for list property
	C[2]	Supports COVP for NULL property value
	C[2]	Supports COVP for BOOLEAN property value
	C[2]	Supports COVP for Enumerated property value
	C[2]	Supports COVP for INTEGER property value
	C[2]	Supports COVP for Unsigned property value
	C[2]	Supports COVP for REAL property value
	C[2]	Supports COVP for Double property value
	C[2]	Supports COVP for Time property value
	C[2]	Supports COVP for Date property value
	C[2]	Supports COVP for CharacterString property value
	C[2]	Supports COVP for OctetString property value
	C[2]	Supports COVP for BitString property value
	C[2]	Supports COVP for BACnetObjectIdentifier property value
	C[2]	Supports COVP for constructed property value
	C[2]	Supports COVP for proprietary property values of basic data types
[1] At least one of these options is required in order to claim conformance to this BIBB. [2] At least one of these options is required in order to claim conformance to this BIBB.		

[In BTL Test Plan, add a section for DS-COVP-B]

4.19 Data Sharing - Change Of Value Property - B

4.19.1 Base Requirements

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

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

BTL - 9.11.1.1 - Confirmed COV Notifications for a SubscribeCOVProperty subscription		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Select parameters for an object and property which supports SubscribeCOVProperty..
	Testing Hints	
	Notes & Results	
BTL - 9.11.1.2 - Unconfirmed COV Notifications for a SubscribeCOVProperty subscription		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Apply the test to an object and property which supports SubscribeCOVProperty..
	Testing Hints	
	Notes & Results	
135.1-2013 - 9.11.1.4 - Canceling COV Subscriptions		
	Test Method	**Manual**
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 9.11.1.5 - Canceling Expired or Non-Existing Subscriptions		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 9.11.1.7 Finite Lifetime Subscriptions		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
135.1-2013 - 9.11.1.8 Updating Existing Subscriptions		
	Test Method	**Manual**
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 9.11.1.9 Client-Supplied COV Increment		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	

	Testing Hints	
	Notes & Results	
BTL- 9.11.2.1 - The Monitored Object Does Not Support COV Notification		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed, unless all objects support SubscribeCOVProperty on at least one of its properties.
	Test Directives	Apply the test to a property in an object that does not support COV (on any property).
	Testing Hints	
	Notes & Results	
BTL- 9.11.2.2 - The Monitored Property Does Not Support COV Notification		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed, unless all objects support SubscribeCOVProperty on all properties.
	Test Directives	Apply the test to a property for which the IUT does not support COV, which is contained in an object that does support COV (on a different property).
	Testing Hints	
	Notes & Results	
BTL- 9.11.2.X11- Monitored Object Does Not Exist		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed if Protocol_Revision >= 15
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL- 9.11.2.X12 - Monitored Property Does Not Exist		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed if Protocol_Revision >= 15
	Test Directives	Be sure to test at least one property identifier that is within the ASHRAE allocated range for standard property identifiers, but that has not yet been defined.
	Testing Hints	
	Notes & Results	
BTL- 9.11.2.X13 - There Is No Space For Subscription		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed
	Test Directives	
	Testing Hints	
	Notes & Results	

BTL- 9.11.2.X14 - The Lifetime Parameter is Out of Range		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed if Protocol_Revision >= 15
	Test Directives	
	Testing Hints	
	Notes & Results	

4.19.2 Supports Lifetimes up to 8 Hours in Duration

The IUT will accept COVP subscriptions with lifetimes up to 8 hours.

BTL - 9.11.1.X10 - Accepts SubscribeCOVProperty-Requests with 8 Hour Lifetimes		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

4.19.3 Supports COVP for Status_Flags changes

The IUT supports change of value notifications for Status_Flags changes

BTL - 9.11.1.X21 Confirmed Change of Value Notification from Status_Flags Property		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed if object type contains a Status_Flag and property which supports SubscribeCOVProperty.
	Test Directives	Repeat test for at least one object of each type that has at least one property which supports SubscribeCOVProperty
	Testing Hints	
	Notes & Results	
BTL - 9.11.1.X22 Unconfirmed Change of Value Notification from Status_Flags Property		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed if object type contains a Status_Flag and property which supports SubscribeCOVProperty.
	Test Directives	Repeat test for at least one object of each type that has at least one property which supports SubscribeCOVProperty
	Testing Hints	
	Notes & Results	

4.19.4 Supports COVP to non-array properties

The IUT supports change of value notifications for at least one non-array property

BTL - 9.11.1.1 - Confirmed COV Notifications for a SubscribeCOVProperty subscription		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Select parameters for an object and property which supports SubscribeCOVProperty..
	Testing Hints	
	Notes & Results	
BTL - 9.11.1.2 - Unconfirmed COV Notifications for a SubscribeCOVProperty subscription		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Select parameters for an object and property which supports SubscribeCOVProperty
	Testing Hints	
	Notes & Results	

4.19.5 Supports COVP to array elements

The IUT supports change of value notifications for at least one array element.

BTL - 9.11.1.1 - Confirmed COV Notifications for a SubscribeCOVProperty subscription		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Select parameters for an object and property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	
BTL - 9.11.1.2 - Unconfirmed COV Notifications for a SubscribeCOVProperty subscription		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Select parameters for an object and property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	

4.19.6 Supports COVP to the size of an array

The IUT supports change of value notifications for at least one index 0 of an array

BTL - 9.11.1.1 - Confirmed COV Notifications for a SubscribeCOVProperty subscription		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.

	Test Conditionality	Must be executed.
	Test Directives	Select parameters for an object and property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	
BTL - 9.11.1.2 - Unconfirmed COV Notifications for a SubscribeCOVProperty subscription		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Select parameters for an object and property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	

4.19.7 Supports COVP to whole arrays

The IUT supports change of value notifications for at least one whole array

BTL - 9.11.1.1 - Confirmed COV Notifications for a SubscribeCOVProperty subscription		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Select parameters for an object and property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	
BTL - 9.11.1.2 - Unconfirmed COV Notifications for a SubscribeCOVProperty subscription		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Select parameters for an object and property which supports SubscribeCOVProperty
	Testing Hints	
	Notes & Results	

4.19.8 Supports COVP to a list property

The IUT supports change of value notifications for at least one list property

BTL - 9.11.1.1 - Confirmed COV Notifications for a SubscribeCOVProperty subscription		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Select parameters for an object and property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	

BTL - 9.11.1.2 - Unconfirmed COV Notifications for a SubscribeCOVProperty subscription		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Select parameters for an object and property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	

4.19.9 Supports COVP to NULL property value

The IUT supports change of value notifications for at least one property value from datatype NULL

BTL - 9.11.1.X11 - Confirmed Change of Value Notification from Property Value		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	
BTL - 9.11.1.X12 - Unconfirmed Change of Value Notification from Property Value		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	

4.19.10 Supports COVP to BOOLEAN property value

The IUT supports change of value notifications for at least one property value from datatype BOOLEAN

BTL - 9.11.1.X11 - Confirmed Change of Value Notification from Property Value		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	
BTL - 9.11.1.X12 - Unconfirmed Change of Value Notification from Property Value		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	

4.19.11 Supports COVP to Enumerated property value

The IUT supports change of value notifications for at least one property value from datatype Enumerated

BTL - 9.11.1.X11 - Confirmed Change of Value Notification from Property Value		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	
BTL - 9.11.1.X12 - Unconfirmed Change of Value Notification from Property Value		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	

4.19.12 Supports COVP to Integer property value

The IUT supports change of value notifications for at least one INTEGER property value from datatype Integer

BTL - 9.11.1.X11 - Confirmed Change of Value Notification from Property Value		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	
BTL - 9.11.1.X12 - Unconfirmed Change of Value Notification from Property Value		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	

4.19.13 Supports COVP to Unsigned property value

The IUT supports change of value notifications for at least one Property value from datatype Unsigned

BTL - 9.11.1.X11 - Confirmed Change of Value Notification from Property Value		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.

	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	
BTL - 9.11.1.X12 - Unconfirmed Change of Value Notification from Property Value		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	

4.19.14 Supports COVP to REAL property value

The IUT supports change of value notifications for at least one property value from datatype real

BTL - 9.11.1.X11 - Confirmed Change of Value Notification from Property Value		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	
BTL - 9.11.1.X12 - Unconfirmed Change of Value Notification from Property Value		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	

4.19.15 Supports COVP to Double property value

The IUT supports change of value notifications for at least one property value from datatype Double

BTL - 9.11.1.X11 - Confirmed Change of Value Notification from Property Value		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	
BTL - 9.11.1.X12 - Unconfirmed Change of Value Notification from Property Value		

	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	

4.19.16 Supports COVP to Time property value

The IUT supports change of value notifications for at least one property value from datatype Time

BTL - 9.11.1.X11 - Confirmed Change of Value Notification from Property Value		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty..
	Testing Hints	
	Notes & Results	
BTL - 9.11.1.X12 - Unconfirmed Change of Value Notification from Property Value		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	

4.19.17 Supports COVP to Date property value

The IUT supports change of value notifications for at least one property value from datatype Date

BTL - 9.11.1.X11 - Confirmed Change of Value Notification from Property Value		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	
BTL - 9.11.1.X12 - Unconfirmed Change of Value Notification from Property Value		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty..
	Testing Hints	
	Notes & Results	

4.19.18 Supports COVP to CharacterString property value

The IUT supports change of value notifications for at least one property value from datatype CharacterString

BTL - 9.11.1.X11 - Confirmed Change of Value Notification from Property Value		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	
BTL - 9.11.1.X12 - Unconfirmed Change of Value Notification from Property Value		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	

4.19.19 Supports COVP to OctetString property value

The IUT supports change of value notifications for at least one property value from datatype OctedString

BTL - 9.11.1.X11 - Confirmed Change of Value Notification from Property Value		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty...
	Testing Hints	
	Notes & Results	
BTL - 9.11.1.X12 - Unconfirmed Change of Value Notification from Property Value		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	

4.19.20 Supports COVP to BitString property value

The IUT supports change of value notifications for at least one property value from datatype BitString

BTL - 9.11.1.X11 - Confirmed Change of Value Notification from Property Value		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one

		property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	
BTL - 9.11.1.X12 - Unconfirmed Change of Value Notification from Property Value		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	

4.19.21 Supports COVP to BACnetObjectIdentifier property value

The IUT supports change of value notifications for at least one property value from datatype BACnetObjectIdentifier

BTL - 9.11.1.X11 - Confirmed Change of Value Notification from Property Value		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	
BTL - 9.11.1.X12 - Unconfirmed Change of Value Notification from Property Value		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty...
	Testing Hints	
	Notes & Results	

4.19.22 Supports COVP to constructed property value

The IUT supports change of value notifications for at least one constructed property value

BTL - 9.11.1.X11 - Confirmed Change of Value Notification from Property Value		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty...
	Testing Hints	
	Notes & Results	
BTL - 9.11.1.X12 - Unconfirmed Change of Value Notification from Property Value		
	Test Method	**Manual**

	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty..
	Testing Hints	
	Notes & Results	

4.19.23 Supports COVP to proprietary property values of basic data types

The IUT supports change of value notifications for at least one proprietary property values of basic data types

BTL - 9.11.1.X11 - Confirmed Change of Value Notification from Property Value		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	
BTL - 9.11.1.X12 - Unconfirmed Change of Value Notification from Property Value		
	Test Method	**Manual**
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test at least once for each object type that has at least one property which supports SubscribeCOVProperty.
	Testing Hints	
	Notes & Results	

[In BTL Specified Tests, derive modified versions of 5 existing tests in 135.1-2013, for DS-COVP-B]

BTL Specified Tests

[In BTL Specified Tests, derive modified versions of 5 existing tests in 135.1-2013, for DS-COVP-B]

9.11.1.1 Confirmed COV Notifications
Reason for Change: Remove the allowance for devices which do not support both confirmed and unconfirmed notifications.

Purpose: To verify that the IUT correctly responds to a SubscribeCOVProperty request to establish a subscription for confirmed COV notifications. ~~An implementation that supports COV reporting cannot respond with an error for both this test and the test in 9.11.1.2.~~

Test Steps:

1. TRANSMIT SubscribeCOVProperty-Request,
   'Subscriber Process Identifier' = (any valid process identifier),
   'Monitored Object Identifier' = (any object supporting COV notifications),
   'Issue Confirmed Notifications' = TRUE,
   'Lifetime' = (any value > 0~~ if automatic cancellation is supported, otherwise 0~~),
   'Monitored Property Identifier' = (any valid property supporting COV notifications)
2. RECEIVE BACnet-SimpleACK-PDU
3. BEFORE **Notification Fail Time**
   ~~IF (the IUT supports confirmed notifications) THEN~~
   RECEIVE BACnetConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = (the same identifier used in the subscription),
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = (the same object used in the subscription),
   'Time Remaining' = (any value > 0~~ if automatic cancellation is supported, otherwise 0~~),
   'List of Values' = (values appropriate to the property subscribed to, and any other properties the IUT provides with it, such as Status-Flags)
   ~~ELSE~~
   ~~RECEIVE BACnet-Error PDU,~~
   ~~Error Class = SERVICES,~~
   ~~Error Code = SERVICE_REQUEST_DENIED | OTHER~~

4. TRANSMIT BACnet-SimpleACK-PDU

9.11.1.2 Unconfirmed COV Notifications
Reason for Change: Remove the allowance for devices which do not support both confirmed and unconfirmed notifications.

Purpose: To verify that the IUT correctly responds to a SubscribeCOVProperty request to establish a subscription for Unconfirmed COV notifications.~~ An implementation that supports COV reporting cannot respond with an error for both this test and the test in 9.11.1.1.~~

Test Steps:

1. TRANSMIT SubscribeCOVProperty-Request,
   'Subscriber Process Identifier' = (any valid process identifier),
   'Monitored Object Identifier' = (any object supporting COV notifications),
   'Issue Confirmed Notifications' = FALSE,
   'Lifetime' = (any value > 0~~ if automatic cancellation is supported, otherwise 0~~),
   'Monitored Property Identifier' = (any valid property supporting COV notifications)
2. RECEIVE BACnet-SimpleACK-PDU
3. BEFORE **Notification Fail Time**

```
~~IF (the IUT supports unconfirmed notifications) THEN~~
    RECEIVE BACnetUnconfirmedCOVNotification-Request,
        'Subscriber Process Identifier' =   (the same identifier used in the subscription),
        'Initiating Device Identifier' =    IUT,
        'Monitored Object Identifier' =     (the same object used in the subscription),
        'Time Remaining' =                  (any value > 0~~ if automatic cancellation is supported, otherwise 0~~),
        'List of Values' =                  (values appropriate to the property subscribed to, and any other properties
                                            the IUT provides with it, such as Status-Flags)
~~ELSE~~
    ~~RECEIVE BACnet-Error PDU,~~
        ~~Error Class  =     SERVICES,~~
        ~~Error Code  =      SERVICE_REQUEST_DENIED | OTHER~~
```

9.11.1.5 Canceling Expired or Non-Existing Subscriptions

Reason for change: Added missing verification that the IUT did not send a COV notification, and removed superfluous note to tester.

Purpose: To verify that the IUT correctly responds to a SubscribeCOVProperty request to cancel a subscription that no longer exists.

Test Steps:

1. TRANSMIT SubscribeCOVProperty-Request,
'Subscriber Process Identifier' = (any unused process identifier or an identifier from a previously
terminated subscription),
'Monitored Object Identifier' = (any unused object or an object from a previously
terminated subscription),
'Monitored Property Identifier' = (any unused property or a property from a previously terminated
subscription)
2. RECEIVE BACnet-SimpleACK-PDU
3. WAIT **Notification Fail Time**
4. MAKE (a change to the monitored object that would cause a COV notification if there were an active subscription)
5. *CHECK(the IUT did not issue a COV notification)*

~~Notes to Tester: The IUT shall not transmit a COV notification message. An error message is not an acceptable response.~~

9.11.1.7 Finite Lifetime Subscriptions

Reason for change: Updates description of 'Time Remaining' and adds validation that this value counts down as expected.

Purpose: To verify that the IUT correctly responds to a SubscribeCOVProperty request to establish a subscription with a

temporary lifetime. Either confirmed or unconfirmed notifications may be used, but at least one of these options must be

supported by the IUT.

Test Steps:

1. TRANSMIT SubscribeCOVProperty-Request,
'Subscriber Process Identifier' = (any valid process identifier),
'Monitored Object Identifier' = (any object supporting COV notifications),
'Issue Confirmed Notifications' = TRUE | FALSE,

```
     'Lifetime' = (a value between 60 seconds and 300 seconds),
     'Monitored Property Identifier' = (any valid property supporting COV notifications)
2. RECEIVE BACnet-SimpleACK-PDU
~~3. BEFORE Notification Fail Time~~
3. IF (the subscription was for confirmed notifications) THEN
   BEFORE Notification Fail Time
      RECEIVE BACnetConfirmedCOVNotification-Request,
           'Subscriber Process Identifier' = (the same identifier used in the subscription),
           'Initiating Device Identifier' = IUT,
           'Monitored Object Identifier' = (the same object used in the subscription),
           'Time Remaining' = (~~the requested subscription lifetime~~ A value approximately equal to, but not greater
                               than, the requested subscription lifetime),
           'List of Values' = (values appropriate to the property subscribed to, and any other
           properties the IUT provides with it, such as Status-Flags)
   TRANSMIT BACnet-SimpleACK-PDU
ELSE
   BEFORE Notification Fail Time
      RECEIVE BACnetUnconfirmedCOVNotification-Request,
           'Subscriber Process Identifier' = (the same identifier used in the subscription),
           'Initiating Device Identifier' = IUT,
           'Monitored Object Identifier' = (the same object used in the subscription),
           e
           'List of Values' = (values appropriate to the property subscribed to, and any other
           properties the IUT provides with it, such as Status-Flags)
4. MAKE (a change to the monitored object that ~~should~~ causes a COV notification)
5. WAIT a period longer than the resolution of the IUT's COV subscription lifetime timer
~~5. BEFORE Notification Fail Time~~
6. IF (the subscription was for confirmed notifications) THEN
    BEFORE Notification Fail Time
      RECEIVE BACnetConfirmedCOVNotification-Request,
           'Subscriber Process Identifier' = (the same identifier used in the subscription),
           'Initiating Device Identifier' = IUT,
           'Monitored Object Identifier' = (the same object used in the subscription),
           'Time Remaining' = (TR: a value greater than 0 and less than the requested subscription lifetime),
           'List of Values' = (values appropriate to the property subscribed to, and any other
           properties the IUT provides with it, such as Status-Flags)
        TRANSMIT BACnet-SimpleACK-PDU
    ELSE
        BEFORE Notification Fail Time
          RECEIVE BACnetUnconfirmedCOVNotification-Request,
                'Subscriber Process Identifier' = (the same identifier used in the subscription),
                'Initiating Device Identifier' = IUT,
                'Monitored Object Identifier' = (the same object used in the subscription),
                'Time Remaining' = (TR: a value greater than 0 and less than the requested subscription
                lifetime),
                'List of Values' = (values appropriate to the object type of the monitored object
                including the changed value that triggered the notification)
7. WAIT a period longer than the resolution of the IUT's COV subscription lifetime timer
8.   MAKE (a change to the monitored object that causes a COV notification)
```

```
9.  IF (the subscription was for confirmed notifications) THEN
        BEFORE Notification Fail Time
            RECEIVE BACnetUnconfirmedCOVNotification-Request,
                'Subscriber Process Identifier' = (the same identifier used in the subscription),
                'Initiating Device Identifier' = IUT,
                'Monitored Object Identifier' = (the same object used in the subscription),
                'Time Remaining' = (a value greater than 0 and less than the TR),
                'List of Values' = (values appropriate to the object type of the monitored object)
    ELSE
        BEFORE Notification Fail Time
            RECEIVE BACnetUnconfirmedCOVNotification-Request,
                'Subscriber Process Identifier' = (the same identifier used in the subscription),
                'Initiating Device Identifier' = IUT,
                'Monitored Object Identifier' = (the same object used in the subscription),
                'Time Remaining' = (a value greater than 0 and less than the TR),
                'List of Values' = (values appropriate to the object type of the monitored object
                including the changed value that triggered the notification)
~~6~~10. WAIT (the lifetime of the subscription)
~~7~~11. MAKE (a change to the monitored object that would cause a COV notification if there were an active subscription)
12. CHECK (verify that the IUT did not transmit a COV notification message)
```

~~Notes to Tester: The IUT shall not transmit a COV notification message addressed to the TD after step 6~~.

9.11.1.9 Client-Supplied COV Increment

Reason for Change: Modify the test to work with all numeric datatypes.

Purpose: To verify that the IUT correctly generates COV notifications when the client supplies the COV increment in the SubscribeCOVProperty request. Either confirmed or unconfirmed notifications may be used but at least one of these options must be supported by the IUT.

Test Concept: A subscription for COV notification is made for a property of *numeric* datatype~~ REAL~~. The subscription request specifies a COV increment. The monitored property is changed by an amount less than the increment, and the TD waits to ensure that the IUT does not generate a notification. The monitored property is changed by an amount slightly more than is required to cause a COV notification, and the TD waits for the notification.

Test Configuration: If the property being subscribed to has a related COV_Increment property in the object, then the value of the COV_Increment property should be significantly different than the COV increment provided in the subscription service.

Test Steps:

```
1.  TRANSMIT SubscribeCOVProperty-Request,
        'Subscriber Process Identifier' =      (any valid process identifier),
        'Monitored Object Identifier' =        (any object supporting COV notifications),
        'Issue Confirmed Notifications' =      TRUE | FALSE,
        'Lifetime' =                           (any value that will ensure no re-subscription is required to complete the test),
        'Monitored Property Identifier' =      (any valid property supporting COV notifications),
        'COV Increment' =                      (any valid increment value)
2.  RECEIVE BACnet-SimpleACK-PDU
3.  BEFORE Notification Fail Time
        IF (the subscription was for confirmed notifications) THEN
            RECEIVE BACnetConfirmedCOVNotification-Request,
            'Subscriber Process Identifier' =   (the same identifier used in the subscription),
            'Initiating Device Identifier' =    IUT,
            'Monitored Object Identifier' =     (the same object used in the subscription),
            'Time Remaining' ~=                 (the requested lifetime),
            'List of Values' =                  (values appropriate to the object type of the monitored object
                                                including the value of monitored property)
        TRANSMIT BACnet-SimpleACK-PDU
    ELSE
        RECEIVE BACnetUnconfirmedCOVNotification-Request,
            'Subscriber Process Identifier' =     (the same identifier used in the subscription),
            'Initiating Device Identifier' =      IUT,
            'Monitored Object Identifier' =       (the same object used in the subscription),
            'Time Remaining' ~=                   (the requested lifetime),
            'List of Values' =                    (values appropriate to the object type of the monitored object
                                                  including the value of monitored property)
4.  MAKE (the monitored property change by less than the COV increment)
5.  WAIT Notification Fail Time
6.  CHECK (verify that the IUT did not transmit a notification message for the monitored property)
7.  MAKE (the monitored property change by slightly more than COV Increment less the amount changed in step
        4)
8.  BEFORE Notification Fail Time
    IF (the subscription was for confirmed notifications) THEN
        RECEIVE BACnetConfirmedCOVNotification-Request,
            'Subscriber Process Identifier' =     (the same identifier used in the subscription),
            'Initiating Device Identifier' =      IUT,
            'Monitored Object Identifier' =       (the same object used in the subscription),
            'Time Remaining' =                    ?,
            'List of Values' =                    (values appropriate to the object type of the monitored object
                                                  including the changed value that triggered the notification)
        TRANSMIT BACnet-SimpleACK-PDU
    ELSE
        RECEIVE BACnetUnconfirmedCOVNotification-Request,
            'Subscriber Process Identifier' =     (the same identifier used in the subscription),
            'Initiating Device Identifier' =      IUT,
            'Monitored Object Identifier' =       (the same object used in the subscription),
            'Time Remaining' =                    ?,
            'List of Values' =                    (values appropriate to the object type of the monitored object
                                                  including the changed value that triggered the notification)
9. TRANSMIT SubscribeCOVProperty-Request,
        'Subscriber Process Identifier' =         (the same identifier used in the subscription),
        'Monitored Object Identifier' =            (the same object used in the subscription)
        'Monitored Property Identifier' =          (the same property used in the subscription)
10.  RECEIVE BACnet-SimpleACK-PDU
```

[In BTL Specified Tests, derive modified versions of two existing tests in 135.1-2013, with specified responses different from those in the modified versions of those tests in 135.1-2013o, for DS-COVP-B]

9.11.1.X10 Accepts SubscribeCOVProperty-Requests with 8 Hour Lifetimes

Purpose: To verify that the IUT correctly accepts lifetimes of at least 8 hours.

Test Steps:

```
1.  TRANSMIT SubscribeCOVProperty-Request,
        'Subscriber Process Identifier' =        (any valid process identifier),
        'Monitored Object Identifier' =          (any object supporting COV notifications),
        'Issue Confirmed Notifications' =        TRUE | FALSE,
        'Lifetime' =                             28800
        'Monitored Property Identifier' =        (any valid property supporting COV notifications)
2.  RECEIVE BACnet-SimpleACK-PDU
3.  BEFORE **Notification Fail Time**
        IF (the subscription was for confirmed notifications) THEN
            RECEIVE BACnetConfirmedCOVNotification-Request,
                'Subscriber Process Identifier' =   (the same identifier used in the subscription),
                'Initiating Device Identifier' =    IUT,
                'Monitored Object Identifier' =     (the same object used in the subscription),
                'Time Remaining' ~=                 (the requested lifetime),
                'List of Values' =                  (values appropriate to the object type of the monitored object
                                                    including the value of monitored property)
            TRANSMIT BACnet-SimpleACK-PDU
        ELSE
            RECEIVE BACnetUnconfirmedCOVNotification-Request,
                'Subscriber Process Identifier' =     (the same identifier used in the subscription),
                'Initiating Device Identifier' =      IUT,
                'Monitored Object Identifier' =       (the same object used in the subscription),
                'Time Remaining' ~=                   (the requested lifetime),
                'List of Values' =                    (values appropriate to the object type of the monitored object
                                                      including the value of monitored property)
4.  TRANSMIT SubscribeCOVProperty-Request,
        'Subscriber Process Identifier' =        (the same identifier used in Step 1),
        'Monitored Object Identifier' =          (the same identifier used in the subscription),
        'Monitored Property Identifier' =        (the same object used in the subscription)

5.  RECEIVE BACnet-SimpleACK-PDU
```

9.11.1.X11 Confirmed Change of Value Notification from Property Value

Purpose: To verify that the IUT can initiate ConfirmedCOVNotification service requests conveying a change of the Property Value.

Test Concept: A property subscription for COV notifications is established, using a Lifetime of L. L shall be set to a value less than 24 hours and large enough to complete the test. The Value of the monitored Property is changed and a notification shall be received. The subscribed property may be changed using the WriteProperty service or by another means. For implementations where it is not possible to write to these properties at all the vendor shall provide an alternative trigger mechanism to accomplish this task. All of these methods are equally acceptable.

Test Steps:

```
1.  TRANSMIT SubscribeCOVProperty-Request,
                'Subscriber Process Identifier' =   (any valid process identifier),
                 'Monitored Object Identifier' =    X
                'Issue Confirmed Notifications' = TRUE,
                'Lifetime' =                        L
                'Monitored Property Identifier' = Y (any valid property supporting COV notifications)
2.  RECEIVE BACnet-SimpleACK-PDU
3.  BEFORE Notification Fail Time
        RECEIVE BACnetConfirmedCOVNotification-Request,
                'Subscriber Process Identifier' =      (the same identifier used in the subscription),
```

'Initiating Device Identifier' = IUT,
'Monitored Object Identifier' = X
'Time Remaining' = (any value appropriate for the Lifetime selected),
'List of Values' = (values appropriate to the property subscribed to, and any other properties the IUT provides with it, such as Status-Flags)

4. TRANSMIT BACnet-SimpleACK-PDU
5. MAKE (a change to the monitored object PROPERTY that causes a COV notification)
6. BEFORE Notification Fail Time
RECEIVE BACnetConfirmedCOVNotification-Request,
'Subscriber Process Identifier' = (the same identifier used in the subscription),
'Initiating Device Identifier' = IUT,
'Monitored Object Identifier' = X
'Time Remaining' = (any value appropriate for the Lifetime selected),
'List of Values' = (values appropriate to the property subscribed to, and any other properties the IUT provides with it, such as Status-Flags)

7. TRANSMIT SubscribeCOVProperty-Request,
'Subscriber Process Identifier' = (the same identifier used in Step 1),
'Monitored Object Identifier' = X
'Monitored Property Identifier' = Y

8. RECEIVE BACnet-SimpleACK-PDU

[In BTL Specified Tests, add new tests for DS-COVP-B, as shown]

9.11.1.X12 Unconfirmed Change of Value Notification from Property Value

Purpose: To verify that the IUT can initiate UnconfirmedCOVNotification service requests conveying a change of the Property Value.

Test Steps: The steps for this test case are identical to the test steps in 9.11.1.X11 except that the SubscribeCOVProperty service request in step 1 shall have a value of FALSE for the 'Issue Confirmed Notifications' parameter, all of the ConfirmedCOVNotification requests shall be UnconfirmedCOVNotification requests, and there is no acknowledgment of the unconfirmed services. The MAC address used for the notification message shall be such that the TD is one of the recipients

9.11.1.X21 Confirmed Change of Value Notification from Status_Flags Property

Purpose: To verify that the IUT can initiate ConfirmedCOVNotification service requests conveying a change of the Status_Flags Property.

Test Concept: A property subscription for COV notifications is established, using a Lifetime of L. L shall be set to a value less than 24 hours and large enough to complete the test. The Status_Flags property of the monitored object is then changed and a notification shall be received. The value of the Status-Flags property can be changed by using the WriteProperty service or by another means. For some implementations writing to the Out_Of_Service property will accomplish this task. For implementations where it is not possible to write to Status_Flags or Out_Of_Service or change the Status_Flags by any other means, this test shall be skipped.

Test Steps:

1. TRANSMIT SubscribeCOVProperty-Request,
'Subscriber Process Identifier' = (any valid process identifier),
'Monitored Object Identifier' = X
'Issue Confirmed Notifications' = TRUE,

```
                    'Lifetime' =                    L
                    'Monitored Property Identifier' =     Y (any valid property but not Status_Flag supporting COV
                                                          notifications)
2.  RECEIVE BACnet-SimpleACK-PDU
3.  BEFORE Notification Fail Time
          RECEIVE BACnetConfirmedCOVNotification-Request,
                    'Subscriber Process Identifier' =      (the same identifier used in the subscription),
                    'Initiating Device Identifier' =       IUT,
                    'Monitored Object Identifier' =        X
                    'Time Remaining' =                     (any value appropriate for the Lifetime selected),
                    'List of Values' =                     (values appropriate to the property subscribed to and initial
                                                           Status_Flags)
4. TRANSMIT BACnet-SimpleACK-PDU
5. MAKE (Status_Flags = any value that differs from "initial Status_Flags")
6. BEFORE Notification Fail Time
          RECEIVE BACnetConfirmedCOVNotification-Request,
                    'Subscriber Process Identifier' =      (the same identifier used in the subscription),
                    'Initiating Device Identifier' =       IUT,
                    'Monitored Object Identifier' =        X
                    'Time Remaining' =                     (any value appropriate for the Lifetime selected),
                    'List of Values' =                     (initial values appropriate to the property subscribed to and
                                                  new Status_Flags)

7. TRANSMIT SubscribeCOVProperty-Request,
               'Subscriber Process Identifier' =           (the same identifier used in Step 1),
               'Monitored Object Identifier' = X
               'Monitored Property Identifier' =           Y

8.  RECEIVE BACnet-SimpleACK-PDU
```

9.11.1.X22 Unconfirmed Change of Value Notification from Status_Flags Property

Purpose: To verify that the IUT can initiate UnconfirmedCOVNotification service requests conveying a change of the Status_Flags Property.

Test Steps: The steps for this test case are identical to the test steps in 9.11.1.X21 except that the SubscribeCOVProperty service request in step 1 shall have a value of FALSE for the 'Issue Confirmed Notifications' parameter, all of the ConfirmedCOVNotification requests shall be UnconfirmedCOVNotification requests, and there is no acknowledgment of the unconfirmed services. The MAC address used for the notification message shall be such that the TD is one of the recipients

[In BTL Specified Tests, derive modified versions of two existing tests in 135.1-2013, with specified responses different from those in the modified versions of those tests in 135.1-2013o, for DS-COVP-B]

9.11.2.1 The Monitored Object Does Not Support COV Notification

Reason for Change: Update the accepted error responses as per changes made in Protocol_Revision 15.

Purpose: To verify that the IUT correctly responds to a SubscribeCOVProperty request to establish a subscription when the monitored object does not support COV notifications.

Test Steps:

```
1.        TRANSMIT SubscribeCOVProperty-Request,
               'Subscriber Process Identifier' =    (any valid process identifier),
```

'Monitored Object Identifier' = (any object that does not support COV notifications),
'Issue Confirmed Notifications' = TRUE,
'Lifetime' = 60,
'Monitored Property Identifier' = (any property in the object)
~~2. RECEIVE BACnet-Error-PDU,~~
~~Error Class = SERVICES,~~
~~Error Code = SERVICE_REQUEST_DENIED | OTHER~~
2. IF (Protocol_Revision is present and Protocol_Revision ≥ 15) THEN
RECEIVE
(BACnet-Error-PDU,
Error Class = OBJECT,
Error Code = OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED) |
(BACnet-Error-PDU,
Error Class = PROPERTY,
Error Code = NOT_COV_PROPERTY)
ELSE
RECEIVE
(BACnet-Error-PDU,
Error Class = SERVICES,
Error Code = SERVICE_REQUEST_DENIED | OTHER) |
(BACnet-Error-PDU,
Error Class = OBJECT,
Error Code = OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED) |
(BACnet-Error-PDU,
Error Class = PROPERTY,
Error Code = NOT_COV_PROPERTY)

9.11.2.2 The Monitored Property Does Not Support COV Notification

Reason for Change: Update the accepted error responses as per changes made in Protocol_Revision 15.

Purpose: To verify that the IUT correctly responds to a SubscribeCOVProperty request to establish a subscription when the monitored object supports COV notifications but not on the requested property.

Test Steps:

1. TRANSMIT SubscribeCOVProperty-Request,
'Subscriber Process Identifier' = (any valid process identifier),
'Monitored Object Identifier' = (any object that supports COV notifications),
'Issue Confirmed Notifications' = TRUE,
'Lifetime' = 60,
'Monitored Property Identifier' = (any property that does not support COV notifications)
~~2. RECEIVE BACnet-Error-PDU,~~
~~Error Class = SERVICES,~~
~~Error Code = SERVICE_REQUEST_DENIED | OTHER~~
2. IF (Protocol_Revision is present and Protocol_Revision ≥ 15) THEN
RECEIVE BACnet-Error-PDU,
Error Class = PROPERTY,
Error Code = NOT_COV_PROPERTY
ELSE
RECEIVE
(BACnet-Error-PDU,
Error Class = SERVICES,
Error Code = SERVICE_REQUEST_DENIED | OTHER) |
(BACnet-Error-PDU,
Error Class = PROPERTY,
Error Code = NOT_COV_PROPERTY)

[In BTL Specified Tests, add new tests for DS-COVP-B, as shown]

9.11.2.X11 Monitored Object Does Not Exist
Purpose: To verify that the IUT correctly responds to a SubscribeCOVProperty request to establish a subscription when the monitored object does not exist.

Test Steps:

```
1.      TRANSMIT SubscribeCOVProperty-Request,
            'Subscriber Process Identifier' =       (any valid process identifier),
            'Monitored Object Identifier' =   (any object of a type that supports COV and an instance which does not
exist in the IUT),
            'Issue Confirmed Notifications' =       TRUE,
            'Lifetime' =                            60
            'Monitored Property Identifier' =       (any valid property supporting COV notifications)
2. RECEIVE BACnet-Error-PDU,
                Error Class = OBJECT,
                Error Code = UNKNOWN_OBJECT
```

9.11.2.X12 Monitored Property Does Not Exist
Purpose: To verify that the IUT correctly responds to a SubscribeCOVProperty request to establish a subscription when the monitored property does not exist.

Test Steps:

```
1.      TRANSMIT SubscribeCOVProperty-Request,
            'Subscriber Process Identifier' =       (any valid process identifier),
            'Monitored Object Identifier' =         (object supporting COV notifications),
            'Issue Confirmed Notifications' =       TRUE,
            'Lifetime' =                            60
            'Monitored Property Identifier' =      (any valid property supporting COV notifications which does
                                                   not exist for specified object)
2. RECEIVE BACnet-Error-PDU,
                Error Class = PROPERTY,
                Error Code = UNKNOWN_PROPERTY
```

9.11.2.X13 There Is No Space For Subscription
Purpose: To verify that the IUT correctly responds to a SubscribeCOVProperty request to establish a subscription when there is no space for a subscription.

Test Concept: Repeatedly subscribe to the same object each time with a different Process Identifier until the device runs out of resources and returns the appropriate error. This test only applies to IUTs that claim a Protocol_Revision of 10 or higher.

Test Conditionality: If the device cannot be configured such that the maximum number of subscriptions the IUT can accept is less than 10000, then this test may be skipped.

Test Steps:

```
REPEAT PID = (1 through the maximum number of subscriptions the IUT can accept plus 1, or until the IUT
returns an Error-PDU) {

1.      TRANSMIT SubscribeCOVProperty-Request,
            'Subscriber Process Identifier' =       PID,
            'Monitored Object Identifier' =         (object supporting COV notifications),
            'Issue Confirmed Notifications' =       TRUE,
          'Lifetime' =                          6000
          'Monitored Property Identifier' =     (any valid property supporting COV notifications)
2. RECEIVE BACnet-SimpleACK-PDU |
              (BACnet-Error-PDU,
                      Error Class = RESOURCES,
                      Error Code = NO_SPACE_TO_ADD_LIST_ELEMENT)
}
```

9.11.2.X14 The Lifetime Parameter is Out of Range

Purpose: To verify that the IUT correctly responds to a SubscribeCOVProperty request to establish a subscription when the Lifetime parameter is out of range.

Test Steps:

```
1.      TRANSMIT SubscribeCOVProperty-Request,
          'Subscriber Process Identifier' =     (any valid process identifier),
          'Monitored Object Identifier' =       (object supporting COV notifications),
          'Issue Confirmed Notifications' =     TRUE,
          'Lifetime' =                          (a value larger than that supported by the IUT),
          'Monitored Property Identifier' =     (any valid property supporting COV notifications)
2.      IF (Protocol_Revision is present and Protocol_Revision ≥ 15) THEN
                RECEIVE BACnet-Error-PDU,
                        Error Class  =     SERVICES,
                        Error Code  =      VALUE_OUT_OF_RANGE
        ELSE
                RECEIVE BACnet-Error-PDU,
                        Error Class = SERVICES,
                        Error Code = VALUE_OUT_OF_RANGE | SERVICE_REQUEST_DENIED | OTHER
                        | (RECEIVE BACnet-Reject-PDU,
                                Reject Reason = PARAMETER_OUT_OF_RANGE)
```

BTL-TP15.0-7.1.0 Tests for the claim of NM-FDR-A

A device claiming NM-FDR-A at any Protocol_Revision shall comply with the following section.

Overview:

Addendum 135-2012*al* added the NM-FDR-A BIBB. This document makes needed changes in the BTL Test Package to provide for claiming the BIBB.

These changes are not contained in any SSPC proposal.

Changes:

[In BTL Checklist, add new Network Management - Foreign Device Registration - A section]

10 Network Management

Support	**Listing**	**Option**
Network Management - Foreign Device Registration - A		
	R	Base Requirements
	BTL-R	Supports configurable BBMD Address
	O	Supports a mode where it transmits a Broadcast at Startup
	O	Supports configurable Time-to-Live

[In BTL Test Plan, add new Network Management - Foreign Device Registration -A sections at end of section 10]

10 Network Management

10.X2 Network Management - Foreign Device Registration - A

These tests are designed for testing the recurring initiation of a Register-Foreign-Device BVLL to the configured BBMD.

10.X2.1 Base Requirements

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

135.1-2013 - 14.8 - Register-Foreign-Device Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 14.9.X1 - Register-Foreign-Device Enable and Disable Test		
	Test Method	Manual

	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 14.9.X2 - Recurring Register-Foreign-Device Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
135.1-2013 - 14.1.6 - Distribute-Broadcast-To-Network		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
135.1-2013 - 14.1.9 - Original-Unicast-NPDU		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 14.1.10 - Forwarded-NPDU (Two-hop Distribution)		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

10.X2.2 Supports configurable BBMD Address

The IUT supports a configurable BBMD Address to which it sends Register-Foreign-Device NPDU.

BTL - 14.9.X3 - BBMD Address Configuration Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.

	Test Directives	
	Testing Hints	
	Notes & Results	

10.X2.3 Supports a mode where it transmits a Broadcast at Startup

The IUT transmits a Broadcast at Startup, which can be observed preceded by the sending of Register-Foreign-Device NPDU, when configured as a Foreign Device.

BTL - 14.9.X4 - Transmits a Broadcast at Startup preceded by Register-Foreign-Device		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

10.X2.4 Supports configurable Time-to-Live

The IUT supports a configurable Time-to-Live which it uses in the Register-Foreign-Device NPDU it sends.

BTL - 14.9.X5 - Time-to-Live Configuration Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

[Add in BTL Specified tests, these five entirely new tests]

14.9.X1 - Register-Foreign-Device Enable and Disable Test
Reason For Change: This tests that the behavior in test 14.8 can be configured by the product end-user.

Purpose: Verify that the option to issue Register-Foreign-Device requests can be configured by the product end-user.

Test Concept: Using a product end-user interface, configure the mode for use of Register-Foreign-Device requests, and then configure the mode to cease use of Register-Foreign-Device requests.

Configuration Requirements: The means by which the product is here configured shall be part of the product's end-user interface. BBMD1 is the TD simulating a correctly functioning BBMD implementation.

Test Steps:

1. MAKE (IUT enter mode for use of Register-Foreign-Device requests)
2. RECEIVE DA = BBMD1,
   Register-Foreign-Device
3. TRANSMIT BVLC-Result,
   ‘Result Code’ = Successful completion
4. MAKE (the IUT not in mode for use of Register-Foreign-Device requests
5. WAIT (more than 31 seconds longer than the 'Time-to-Live' parameter used in Register-Foreign-Device requests)

6. CHECK (that the IUT did not send any Register-Foreign-Device requests)

14.9.X2 Recurring Register-Foreign-Device Test

Reason For Change: This tests in continuous manner what 14.9.1 tests just once.

Purpose: Verify that mode for use of Register-Foreign-Device repeats the Registration recurringly, when in that mode.

Test Concept: IUT is put in a mode to use Register-Foreign-Device requests, and it is observed that Register-Foreign-Device requests are sent sufficiently frequently to prevent expiration of the registration at the BBMD.

Configuration Requirements: The product's setting of 'BBMD Address' parameter is configured as BBMD1. BBMD1 is the TD simulating a correctly functioning BBMD implementation.

Test Steps:

```
1. MAKE (IUT enter mode for use of Register-Foreign-Device requests)
2. RECEIVE DA = BBMD1,
            Register-Foreign-Device
3. TRANSMIT BVLC-Result,
            ‘Result Code’ = Successful completion
4. BEFORE (the time configured for the 'Time-to-Live' parameter used for Register-Foreign-Device requests)
       RECEIVE DA = BBMD1,
            Register-Foreign-Device
5. TRANSMIT BVLC-Result,
            ‘Result Code’ = Successful completion
6. BEFORE (the time configured for the 'Time-to-Live' parameter used for Register-Foreign-Device requests)
       RECEIVE DA = BBMD1,
            Register-Foreign-Device
7. TRANSMIT BVLC-Result,
            ‘Result Code’ = Successful completion
```

Notes to Tester: There is no need for the recurring request to be sent any more quickly than precisely the 'Time-to-Live' since the standard mandates that the BBMD preserve the registration for 30 seconds past the 'Time-to-Live'.

14.9.X3 BBMD Address Configuration Test

Reason For Change: This tests that the behavior in test 14.8 can be configured by the product end-user.

Purpose: Verify that the parameter in Register-Foreign-Device in test 14.8 can be configured by the product end-user.

Test Concept: Using a product end-user interface, configure the 'BBMD Address' parameter that is used in Register-Foreign-Device requests.

Configuration Requirements: The means by which the product is configured for a 'BBMD Address' can be anything in the product's end-user interface. BBMD1 is the TD simulating a correctly functioning BBMD implementation.

Test Steps:

```
1. MAKE (through the product's end-user interface, the setting of 'BBMD Address' parameter equal BBMD1)
2. MAKE (IUT enter mode for use of Register-Foreign-Device requests)
3. RECEIVE DA = BBMD1,
            Register-Foreign-Device
4. TRANSMIT BVLC-Result,
            ‘Result Code’ = Successful completion
```

14.9.X4 Transmits a Broadcast at Startup preceded by Register-Foreign-Device

Reason For Change: This tests in the specific case of startup, what test 14.9.1expects to observe during ordinary ongoing operation.

Purpose: Verify that mode for use of Register-Foreign-Device and setting of 'BBMD Address' parameter are persistent across reset, and that the issuance of Register-Foreign-Device precedes the first issuance of any broadcast, when in that mode.

Test Concept: IUT is put in a mode to use Register-Foreign-Device requests, persistently so it will be re-established, then IUT is reset, and the timing of Register-Foreign-Device request to re-establish that precedes the first issuance of any broadcast.

Configuration Requirements: The product's setting of 'BBMD Address' parameter is configured as BBMD1. BBMD1 is the TD simulating a correctly functioning BBMD implementation.

Test Steps:

1. MAKE (IUT enter mode for use of Register-Foreign-Device requests, persistently so it will be re-established after any reset)
2. MAKE (IUT reset)
3. RECEIVE DA = BBMD1,
   Register-Foreign-Device
4. TRANSMIT BVLC-Result,
   ‘Result Code’ = Successful completion
5. RECEIVE DA = BBMD1,
   Distribute-Broadcast-To-Network,
   NPDU = (any broadcast)
6. TRANSMIT BVLC-Result,
   ‘Result Code’ = Successful completion

Notes to Tester: For the I-Am, one can precede the Register-Foreign-Device command, as long as then after the Register-Foreign-Device occurs, it is followed by a Distribute-Broadcast-To-Network again, of that I-Am.

14.9.X5 Time-to-Live Configuration Test

Reason For Change: Adds verification that the behavior in test 14.8 can be configured by the product end-user.

Purpose: Verify that the parameter in Register-Foreign-Device in test 14.8 can be configured by the product end-user.

Test Concept: Using a product end-user interface, configure the 'Time-to-Live' parameter that is used in Register-Foreign-Device requests.

Configuration Requirements: The means by which the product is configured can be anything in the product's end-user interface. BBMD1 is the TD simulating a correctly functioning BBMD implementation.

Test Steps:

1. MAKE (through the product's end-user interface, the setting of 'Time-to-Live' parameter equal 120)
2. MAKE (IUT enter mode for use of Register-Foreign-Device requests)
3. RECEIVE DA = BBMD1,
   Register-Foreign-Device,
   'Time-to-Live' = 120
4. TRANSMIT BVLC-Result,
   ‘Result Code’ = Successful completion

BTL-TP15.0-8.1.0 Tests for the claim of GW-EO-B

A device claiming GW-EO-B at any Protocol_Revision shall comply with the following section.

Overview:

Addendum 135-2012*al* added the GW-EO-B BIBB definition. This document makes needed changes in the BTL Test Package to claim the GW-EO-B BIBB.

These changes are not contained in any SSPC proposal.

[In BTL Checklist, add two Optional sections, and remove the footnote in Gateway - Embedded Objects - B tests in section 11 2]

11 Gateway

Gateway - Embedded Objects - B		
	R[1]	Base Requirements
	O	Supports writes that affect values in "gatewayed" devices
	O	Supports Command Prioritization
~~[1]Contact BTL for interim tests for this BIBB.~~		

[In BTL Test Plan, add Gateway - Embedded Objects - B tests in section 11 2]

11 Gateway

. . .

11.2 Gateway - Embedded Objects - B

11.2.1 Base Requirement

Base requirements must be met by any IUT that claims GW-EO-B.

BTL - 9.18.1.X8 - ReadProperty gateway object when non-BACnet device offline		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***
	Test Conditionality	Must be executed.
	Test Directives	The test shall be conducted upon an object which is representing information arriving through a Gateway.
	Testing Hints	
	Notes & Results	
BTL - 9.20.1.X9 - ReadPropertyMultiple gateway object when non-BACnet device offline		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***
	Test Conditionality	If IUT does not support ReadPropertyMultiple service then this test shall be skipped.
	Test Directives	The test shall be conducted upon an object which is representing information arriving through a Gateway.

	Testing Hints	
	Notes & Results	
BTL - 9.21.1.X10 - ReadRange gateway object when non-BACnet device offline		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***
	Test Conditionality	If IUT does not support ReadRange service then this test shall be skipped. If IUT support ReadRange service but does not support list property that maps on to non-BACnet devices, this test shall be skipped.
	Test Directives	The test shall be conducted upon an object which is representing information arriving through a Gateway.
	Testing Hints	
	Notes & Results	

11.2.2 Supports writes that affect values in “gatewayed” devices

The IUT supports DS-WP-B to write values to “gatewayed” devices.

Verify Checklist		
	Test Method	Manual
	Configuration	
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-WP-B.
	Testing Hints	
	Notes & Results	
BTL - 9.22.1.X11 - WriteProperty gateway object when non-BACnet device offline		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***
	Test Conditionality	Must be executed if an object which is representing information arriving through a Gateway contains any writable properties
	Test Directives	The test shall be conducted upon an object which is representing information arriving through a Gateway.
	Testing Hints	
	Notes & Results	
BTL - 9.23.1.X12 - WritePropertyMultiple gateway object when non-BACnet device offline		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***
	Test Conditionality	If IUT does not support WritePropertyMultiple service then this test shall be skipped. Execute this test if an object which is representing information arriving through a Gateway contains any writable properties.
	Test Directives	The test shall be conducted upon an object which is representing information arriving through a Gateway.
	Testing Hints	
	Notes & Results	

11.2.3 Supports Command Prioritization

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

135.1-2013 - 7.3.1.2 - Relinquish Default Test		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	Must be executed.
	Test Directives	The test shall be conducted upon an object which is representing information arriving through a Gateway. If no object can be made to meet the configuration requirements, this test shall be skipped.
	Testing Hints	
	Notes & Results	
135.1-2013 - 7.3.1.3 - Command Prioritization Test		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	Must be executed.
	Test Directives	The test shall be conducted upon an object which is representing information arriving through a Gateway.
	Testing Hints	
	Notes & Results	

[In BTL Specified Test add four new tests as shown, each appended to the section for tests of the service.]

9.18.1.X8 ReadProperty service when non-BACnet device offline

Purpose: To verify that ReadProperty Service executes successfully when non-BACnet device is offline or not in communication.

Test Concept: Object1 is an object which is representing information arriving through a Gateway. P1 is a property in Object1.

Test Steps:

```
1.   CHECK (any vendor-specified indication, that the non-BACnet device is offline)
2.   TRANSMIT ReadProperty Request,
         'Object Identifier' = Object1,
         'Property Identifier' = P1
3.   (RECEIVE BACnet-Abort-PDU,
         'Abort Reason' = APPLICATION_EXCEEDED_REPLY_TIME) |
    (RECEIVE ReadProperty-ACK,
         'Object Identifier' = Object1,
         'Property Identifier' = P1,
         'Property Value' = (V, any valid value))
```

9.20.1.X9 ReadPropertyMultiple gateway object when non-BACnet device offline

Purpose: To verify that ReadPropertyMultiple Service executes successfully and needs only access to the local object, or returns the appropriate error, when the gateway to the non-BACnet device is offline or not in communication.

Test Concept: Object1 is an object which is representing information arriving through a Gateway. P1 is a property in Object1.

Configuration Requirement: The non-BACnet device is not connected to the gateway and the gateway knows that the device is offline.

Test Steps:

```
1.  CHECK (any vendor-specified indication, that the non-BACnet device is offline)
2.  TRANSMIT ReadPropertyMultiple-Request,
        'Object Identifier' = Object1,
        'Property Identifier' = P1
3.  (RECEIVE BACnet-Abort-PDU,
        'Abort Reason' = APPLICATION_EXCEEDED_REPLY_TIME) |
    (RECEIVE ReadPropertyMultiple-ACK,
        'Object Identifier' = Object1,
        'Property Identifier' = P1,
        'Property Value' = (any valid value))
```

9.21.1.X10 ReadRange gateway object when non-BACnet device offline

Purpose: To verify that ReadRange Service executes successfully and needs only access to the local object, or returns the appropriate error, when the gateway to the non-BACnet device is offline or not in communication.

Test Concept: Object1 is an object which is representing information arriving through a Gateway. P1 is a property in Object1.

Configuration Requirement: The non-BACnet device is not connected to the gateway and the gateway knows that the device is offline.

Test Steps:

```
1.  CHECK (any vendor-specified indication, that the non-BACnet device is offline)
2.  TRANSMIT ReadRange-Request,
        'Object Identifier' = Object1,
        'Property Identifier' = P1
3.  (RECEIVE BACnet-Abort-PDU,
        'Abort Reason' = APPLICATION_EXCEEDED_REPLY_TIME) |
    (RECEIVE ReadRange-ACK,
        'Object Identifier' = Object1,
        'Property Identifier' = P1,
        'Property Value' = (any valid value))
```

9.22.1.X11 WriteProperty gateway object when non-BACnet device offline

Purpose: To verify that WritePropertyMultiple Service executes successfully and needs only access to the local object, or returns the appropriate error, when the gateway to the non-BACnet device is offline or not in communication.

Test Concept: Object1 is an object which is representing information arriving through a Gateway. P1 is a property in Object1.

Configuration Requirement: The non-BACnet device is not connected to the gateway and the gateway knows that the device is offline.

Test Steps:

```
1.  CHECK (any vendor-specified indication, that the non-BACnet device is offline)
2.  TRANSMIT WriteProperty-Request,
        'Object Identifier' = Object1,
        'Property Identifier' = P1,
        'Property Value' = (any valid value)
3.  (RECEIVE BACnet-Abort-PDU,
        'Abort Reason' = APPLICATION_EXCEEDED_REPLY_TIME) |
    (RECEIVE Simple-ACK)
```

9.23.1.X12 WritePropertyMultiple gateway object when non-BACnet device offline

Purpose: To verify that WritePropertyMultiple Service executes successfully and needs only access to the local object, or returns the appropriate error, when the gateway to the non-BACnet device is offline or not in communication.

Test Concept: Object1 is an object which is representing information arriving through a Gateway. P1 is a property in Object1.

Configuration Requirement: The non-BACnet device is not connected to the gateway and the gateway knows that the device is offline.

Test Steps:

```
1.  CHECK (any vendor-specified indication, that the non-BACnet device is offline)
2.  TRANSMIT WritePropertyMultiple-Request,
        'Object Identifier' = Object1,
        'Property Identifier' = P1
3.  (RECEIVE BACnet-Abort-PDU,
        'Abort Reason' = APPLICATION_EXCEEDED_REPLY_TIME) |
    (RECEIVE Simple-ACK)
```

BTL-TP15.0-9.1.0: Life Safety Point object

Devices claiming support for a Life Safety Point object must claim support for Protocol_Revision 2 or higher and comply with the following section.

Overview:

Addendum 135-1995*c* added the Life Safety Point object. This document makes needed changes in the BTL Test Package to claim Life Safety Point object.

These changes are not contained in any SSPC proposal.

[In BTL Checklist, add Life Safety Point object type to Section 3, Objects]

Support	Listing	Option
Life Safety Point Object		
	R	Base Requirements
	S	Supports writable Out_Of_Service properties
	O	Supports writable Member_Of property
	O	Contains an object with Reliability_Evaluation_Inhibit Property

[In BTL Test Plan, add Life Safety Point object tests in section 3.X50. In the following addition of new clauses of the Test Plan, these are indicated as entirely new sections verbatim, with plain text, verbatim **bold**, or verbatim ***bold-italic*** as shown.]

3.X50 Life Safety Point Object

3.X50.1 Base Requirements

Base requirements must be met by any IUT that can contain Life Safety Point objects.

BTL-7.3.2.15.X6 – Supports writable Mode property		
	Test Method	Manual
	Configuration	The test shall be executed using a Life Safety Point and Life Safety Zone objects.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL-7.3.2.15.X5 – Support writable Tracking_Value		
	Test Method	Manual
	Configuration	The test shall be executed using a Life Safety Point and Life Safety Zone objects.
	Test Conditionality	If Out_Of_Service can be made TRUE, this test must be executed.
	Test Directives	

	Testing Hints	
	Notes & Results	

BTL-7.3.2.15.X9 – Support Silenced property		
	Test Method	Manual
	Configuration	The test shall be executed using a Life Safety Point and Life Safety Zone objects.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

BTL-7.3.2.15.X7 – Support Operation_Expected property		
	Test Method	Manual
	Configuration	As per BTL Specified Tests. The test shall be executed using a Life Safety Point and Life Safety Zone objects.
	Test Conditionality	If IUT is capable of generating event notifications then, it Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X50.2 Supports writable Out_Of_Service properties

The Out_Of_Service property in Life Safety objects contained in the IUT are writable.

135.1-2013 - 7.3.1.1 - Out_Of_Service, Status_Flags, and Reliability Tests		
	Test Method	Manual
	Configuration	The test shall be executed using a Life Safety Point and Life Safety Zone objects.
	Test Conditionality	If Out_Of_Service can be made TRUE, this test must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X50.3 Support writable Member_Of property

BTL-7.3.2.15.X8 – Support Writable Member_Of property		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X50.4 Contains an object with Reliability_Evaluation_Inhibit property

The IUT contains or can be made to contain a Reliability_Evaluation_Inhibit property that is configurable to a value of TRUE.

BTL - 7.3.1.X8.1 - Reliability_Evaluation_Inhibit Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If no object exists in the IUT for which fault conditions can be generated then this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.1.X8.2 - Reliability_Evaluation_Inhibit Summarization Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If no object exists in the IUT for which fault conditions can be generated then this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	

BTL-TP15.0-9.2.0 Life Safety Zone object

A device claiming the Life Safety Zone object at Protocol_Revision 2 or higher shall comply with the following section.

Overview:

Addendum 135-1995*c* added the Life Safety Zone object type. This document makes needed changes in the BTL Test Package to claim the Life Safety Zone object.

These changes are not contained in any SSPC proposal.

[In BTL Checklist, add Life Safety Zone object type to Section 3, Objects]

Support	Listing	Option
Life Safety Zone Object		
	R	Base Requirements
	S	Supports writable Out_Of_Service properties
	O	Supports writable Member_Of property
	O	Contains an object with Reliability_Evaluation_Inhibit Property

[In BTL Test Plan, add Life Safety Zone object tests in section 3.X51. In the following addition of new clauses of the Test Plan, these are indicated as entirely new sections verbatim, with plain text, verbatim **bold**, or verbatim ***bold-italic*** as shown.]

3.X51 Life Safety Zone Object

3.X51.1 Base Requirements

Base requirements must be met by any IUT that can contain Life Safety Zone objects.

BTL-7.3.2.15.X6 – Supports writable Mode property		
	Test Method	Manual
	Configuration	The test shall be executed using a Life Safety Point and Life Safety Zone objects.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL-7.3.2.15.X5 – Support writable Tracking_Value		
	Test Method	Manual
	Configuration	The test shall be executed using a Life Safety Point and Life Safety Zone objects.
	Test Conditionality	If Out_Of_Service can be made TRUE, this test must be executed.
	Test Directives	
	Testing Hints	

	Notes & Results	

BTL-7.3.2.15.X9 – Support Silenced property		
	Test Method	Manual
	Configuration	The test shall be executed using a Life Safety Point and Life Safety Zone objects.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

BTL-7.3.2.15.X7 – Support Operation_Expected property		
	Test Method	Manual
	Configuration	As per BTL Specified Tests. The test shall be executed using a Life Safety Point and Life Safety Zone objects.
	Test Conditionality	If IUT is capable of generating event notifications then, it Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X51.2 Supports writable Out_Of_Service properties

The Out_Of_Service property in Life Safety objects contained in the IUT are writable.

135.1-2013 - 7.3.1.1 - Out_Of_Service, Status_Flags, and Reliability Tests		
	Test Method	Manual
	Configuration	The test shall be executed using a Life Safety Point and Life Safety Zone objects.
	Test Conditionality	If Out_Of_Service can be made TRUE, this test must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X51.3 Support writable Member_Of property

BTL-7.3.2.15.X8 – Support Writable Member_Of property		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X51.4 Contains an object with Reliability_Evaluation_Inhibit property

The IUT contains or can be made to contain a Reliability_Evaluation_Inhibit property that is configurable to a value of TRUE.

BTL - 7.3.1.X8.1 - Reliability_Evaluation_Inhibit Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If no object exists in the IUT for which fault conditions can be generated then this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.1.X8.2 - Reliability_Evaluation_Inhibit Summarization Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If no object exists in the IUT for which fault conditions can be generated then this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	

BTL-TP15.0-9.3.0 Tests for the claim of AE-LS-A

A device claiming AE-LS-A at Protocol_Revision 2 or higher shall comply with the following section.

Overview:

Addendum 135-1995*c* added the LifeSafetyOperation service. This document makes needed changes in the BTL Test Package to claim the AE-LS-A BIBB.

These changes adapt and extend some existing tests defined in 135.1.

Alarm and Event Management - Life Safety - A

	R	Base Requirements
	R	Initiates LifeSafetyOperation requests
	R	Executes ConfirmedEventNotifications
	R	Executes UnconfirmedEventNotifications
	R	Processes intrinsically generated notifications
	R	Processes algorithmically generated notifications
	R	Processes event notifications with timestamps of the BACnetDateTime form
	R	Processes event notifications with timestamps of the Time form
	R	Processes event notifications with timestamps of the Sequence Number form
	R	Supports AE-ACK-A
	R	Supports AE-AS-A

[In BTL Test Plan, add Alarm and Event Management - Life Safety - A in section 5.22. In the following addition of new clauses of the Test Plan, these are indicated as entirely new sections verbatim, with plain text, verbatim **bold**, or verbatim ***bold-italic*** as shown.]

5.22 Alarm and Event Management - Life Safety - A

5.22.1 Base Requirements

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

BTL - 9.4.X1 - Unsupported Message Text Character Set ConfirmedEventNotification Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If the IUT supports all character sets, this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 9.5.X1 - Unsupported Message Text Character Set UnconfirmedEventNotification Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If the IUT supports all character sets, this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	

5.22.2 Initiates LifeSafetyOperation requests

135.1-2013 - 8.9.1 - LifeSafetyOperation Service Initiation Tests to an Object		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

135.1-2013 - 8.9.2 - LifeSafetyOperation Service Initiation Tests to all Objects in a Device		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

5.22.3 Executes ConfirmedEventNotifications

The IUT is capable of executing ConfirmedEventNotifications with an Event Type of CHANGE_OF_LIFE_SAFETY. This functionality will be covered by the testing of the individual algorithms.

No Specific Test		
	Test Method	
	Configuration	
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT's EPICS claims that it supports the ConfirmedEventNotification service.
	Testing Hints	
	Notes & Results	

5.22.4 Executes UnconfirmedEventNotifications

The IUT is capable of executing UnconfirmedEventNotifications with an Event Type of CHANGE_OF_LIFE_SAFETY. There are currently no tests defined for this functional item.

No Specific Test		
	Test Method	
	Configuration	
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT's EPICS claims that it supports the UnconfirmedEventNotification service.
	Testing Hints	
	Notes & Results	

5.22.5 Processes Intrinsically Generated Notifications

The IUT is capable of executing ConfirmedEventNotifications with an Event Type of CHANGE_OF_LIFE_SAFETY that reference an object type other than Event Enrollment.

135.1-2013 - 9.4.1 - ConfirmedEventNotification Using the Time Form of the 'Timestamp' Parameter and Conveying a Text Message, **135.1-2013 - 9.4.2 - ConfirmedEventNotification Using the DateTime Form of the 'Timestamp' Parameter and no Text Message, or** **135.1-2013 - 9.4.3 - ConfirmedEventNotification Using the Sequence Number Form of the 'Timestamp' Parameter and no Text Message**		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	At least one of the tests must be executed with the Event Object Identifier referencing a BACnet object other than an Event Enrollment object.
	Test Directives	
	Testing Hints	
	Notes & Results	

5.22.6 Processes Algorithmically Generated Notifications

The IUT is capable of executing ConfirmedEventNotifications with an Event Type of CHANGE_OF_LIFE_SAFETY that reference an Event Enrollment object.

135.1-2013 - 9.4.1 - ConfirmedEventNotification Using the Time Form of the 'Timestamp' Parameter and Conveying a Text Message, **135.1-2013 - 9.4.2 - ConfirmedEventNotification Using the DateTime Form of the 'Timestamp' Parameter and no Text Message, or** **135.1-2013 - 9.4.3 - ConfirmedEventNotification Using the Sequence Number Form of the 'Timestamp' Parameter and no Text Message**		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	At least one of the tests must be executed with the Event Object Identifier referencing an Event Enrollment object.
	Test Directives	
	Testing Hints	
	Notes & Results	

5.22.7 Processes Event Notifications with Timestamps of the BACnetDateTime Form

The IUT is capable of executing ConfirmedEventNotifications that contain a timestamp of the BACnetDateTime form.

135.1-2013 - 9.4.2 - ConfirmedEventNotification Using the DateTime Form of the 'Timestamp' Parameter and no Text Message		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

5.22.8 Processes Event Notifications with Timestamps of the Time Form

The IUT is capable of executing ConfirmedEventNotifications that contain a timestamp of the Time form.

135.1-2013 - 9.4.1 - ConfirmedEventNotification Using the Time Form of the 'Timestamp' Parameter and Conveying a Text Message		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

5.22.9 Processes Event Notifications with Timestamps of the Sequence Number Form

The IUT is capable of executing ConfirmedEventNotifications that contain a timestamp of the Sequence Number form.

135.1-2013 - 9.4.3 - ConfirmedEventNotification Using the Sequence Number Form of the 'Timestamp' Parameter and no Text Message		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

5.22.10 Supports AE-ACK-A

The IUT must support AE-ACK-A if it claims support for AE-LS-A.

Verify Checklist		
	Test Method	Manual
	Configuration	
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-ACK-A in the Checklist.
	Testing Hints	
	Notes & Results	

5.22.11 Supports AE-AS-A

The IUT must support AE-AS-A if it claims support for AE-LS-A.

Verify Checklist		
	Test Method	Manual
	Configuration	
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-AS-A in the Checklist.
	Testing Hints	
	Notes & Results	

BTL-TP15.0-9.4.0 Tests for the claim of AE-LS-B

A device claiming AE-LS-B at Protocol_Revision 2 or higher shall comply with the following section.

Overview:

Addendum 135-1995*c* added the LifeSafetyOperation service. This document makes needed changes in the BTL Test Package to claim the AE-LS-B BIBB.

These changes adapt and extend some existing tests defined in 135.1.

[In BTL Test Plan, add Alarm and Event Management - Life Safety - A in section 5.23. In the following addition of the Test Plan, these are indicated as entirely new sections verbatim, with plain text, verbatim **bold**, or verbatim ***bold-italic*** as shown.]

5 Alarm and Event Management BIBBs

Alarm and Event Management - Life Safety - B

	R	Base Requirements
	R	Supports the Notification Class Object
	R	Supports AE-INFO-B
	C[1]	Implements intrinsic alarming in a Life Safety object
	C[1]	Supports the CHANGE_OF_LIFE_SAFETY algorithm in Event_Parameters
	C[2]	Supports AE-ACK-B
	C[3]	Generates event notifications with timestamps of the BACnetDateTime form
	C[3]	Generates event notifications with timestamps of the Sequence Number form
	O	Mode Transition Tests when Event State is Maintained
	O	Supports Event_Message_Texts property
	O	Supports Event_Message_Texts_Config property
[1] At least one of these options must be supported to claim support for this BIBB. [2] Required if EventNotifications with service parameter AckRequired = True can be issued. [3] At least one of these options must be supported to claim support for this BIBB. The BACnetDateTime form of the timestamp is the recommended option.		

5.23 Alarm and Event Management - Life Safety - B

5.23.1 Base Requirements

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

BTL - 7.3.1.10.1 - Event_Enable Tests for TO_OFFNORMAL and TO_NORMAL		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If the IUT cannot be configured to meet the configuration requirements then this test shall be skipped.
	Test Directives	If Event Enrollment objects are supported, ensure this functionality is tested on Event Enrollment objects.
	Testing Hints	The BTL will apply this to a single object. The pretester should apply it to all objects that support alarm generation.
	Notes & Results	

135.1-2013 - 7.3.1.12 - Notify_Type Test		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	If the IUT cannot be configured to meet the 135.1-2013 configuration requirements then this test shall be skipped.
	Test Directives	If Event Enrollment objects are supported, ensure this functionality is tested on Event Enrollment objects.
	Testing Hints	
	Notes & Results	
135.1-2013 - 8.4.8 - CHANGE_OF_LIFE_SAFETY Tests		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	Must be executed. Any of the 8.4.8 tests can be used to ensure that the IUT properly generates ConfirmedEventNotification requests. The specific tests that can be executed are detailed under the test cases for the specific algorithms. As long as one of the tests is executed using ConfirmedEventNotifications, then this test case shall be satisfied.
	Test Directives	
	Testing Hints	
	Notes & Results	
135.1-2013 - 8.5.8 - CHANGE_OF_LIFE_SAFETY TESTS		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	Must be executed. Any of the 8.5.8 tests can be used to ensure that the IUT properly generates UnconfirmedEventNotification requests. The specific tests that can be executed are detailed under the test cases for the specific algorithms. As long as one of the tests is executed using UnconfirmedEventNotifications, then this test case shall be satisfied.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.1.X9.1 - Event_Detection_Enable Inhibits Event Generation		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If Protocol_Revision < 13, then this test shall be skipped.
	Test Directives	The objects selected by the tester should include all variants that differ in the set of supported alarming properties, or the writability of any of those properties. At least one instance of each variant should be selected.
	Testing Hints	
	Notes & Results	
BTL - 7.3.1.X9.2 - Event_Detection_Enable Inhibits FAULT		
	Test Method	Manual

Configuration	As per ***BTL Specified Tests***.
Test Conditionality	If Protocol_Revision < 13, then this test shall be skipped.
Test Directives	The objects selected by the tester should include all variants that differ in the set of supported alarming properties, or the writability of any of those properties. At least one instance of each variant should be selected.
Testing Hints	
Notes & Results	
BTL - 7.3.1.X6.1 - Event_Algorithm_Inhibit Test	
Test Method	Manual
Configuration	As per ***BTL Specified Tests***.
Test Conditionality	If the IUT has no object in which the Event_Algorithm_Inhibit property is present and does not support the Event_Algorithm_Inhibit_Ref property, or has no object in which Event_Detection_Enable can be made TRUE, this test shall be skipped. If the IUT cannot be configured to contain any object capable of an event transition, then this test shall be skipped.
Test Directives	The object types selected by the tester should include all variants that differ in the set of supported alarming properties, or the writability of any of those properties. At least one instance of each variant should be selected.
Testing Hints	
Notes & Results	
BTL - 7.3.1.X7.1 - Event_Algorithm_Inhibit_Ref Test	
Test Method	Manual
Configuration	As per ***BTL Specified Tests***.
Test Conditionality	If the IUT has no object in which the Event_Algorithm_Inhibit_Ref property is present or has no object in which Event_Detection_Enable can be made TRUE, this test shall be skipped.
Test Directives	The object types selected by the tester should include all variants that differ in the set of supported alarming properties, or the writability of any of those properties. At least one instance of each variant should be selected.
Testing Hints	
Notes & Results	
BTL - 7.3.1.X7.2 - Event_Algorithm_Inhibit Writable Test	
Test Method	Manual
Configuration	As per ***BTL Specified Tests***.
Test Conditionality	If the IUT has no object in which the Event_Algorithm_Inhibit_Ref property is absent or can be made uninitialized or has no object in which Event_Detection_Enable can be made TRUE, this test shall be skipped.
Test Directives	The object types selected by the tester should include all variants that differ in the set of supported alarming properties, or the writability of any of those properties. At least one instance of each variant should be selected.
Testing Hints	
Notes & Results	
135.1-2013 - 9.9.1 - Reset Single Object Execution Tests	
Test Method	Manual

	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
135.1-2013 - 9.9.2 - Reset Multiple Object Execution Test		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
135.1-2013 - 9.9.3 - Silencing Execution Test		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

5.23.2 Supports the Notification Class Object

The IUT supports the Notification Class object in order to send notifications.

Verify Checklist		
	Test Method	Manual
	Configuration	
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for the Notification Class Object in the Checklist.
	Testing Hints	
	Notes & Results	

5.23.3 Supports AE-INFO-B

The IUT must support AE-INFO-B if it claims support for AE-N-I-B.

Verify Checklist		
	Test Method	Manual
	Configuration	
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-INFO-B in the Checklist.
	Testing Hints	
	Notes & Results	

5.23.4 Implements Intrinsic Alarming in a Life Safety object

The IUT contains, or can be made to contain, an object other than an Event Enrollment object that can generate CHANGE_OF_LIFE_SAFETY ConfirmedEventNotifications and UnconfirmedEventNotifications.

Verify Checklist		
	Test Method	Manual
	Configuration	
	Test Conditionality	
	Test Directives	This functionality will be tested by the clause 8.4.8 or 8.5.8 tests in that section.
	Testing Hints	
	Notes & Results	

5.23.5 Supports the CHANGE_OF_LIFE_SAFETY algorithm in Event_Parameters

The IUT contains, or can be made to contain an Event Enrollment object that can generate CHANGE_OF_LIFE_SAFETY ConfirmedEventNotifications and UnconfirmedEventNotifications.

Verify Checklist		
	Test Method	Manual
	Configuration	
	Test Conditionality	
	Test Directives	Ensure this functionality is tested on Event Enrollment objects by the clause 8.4.8 or 8.5.8 tests in that section.
	Testing Hints	
	Notes & Results	

5.23.6 Supports AE-ACK-B

The IUT supports AE-ACK-B in order to execute the AcknowledgeAlarm Service if the IUT is able to send initiates EventNotifications with service parameter AckRequired = True.

Verify Checklist		
	Test Method	Manual
	Configuration	
	Test Conditionality	
	Test Directives	If the IUT cannot be configured to contain any object with an unacknowledged event, then this test shall be skipped.
	Testing Hints	
	Notes & Results	

5.23.7 Generates Event Notifications with Timestamps of the BACnetDateTime Form

The IUT generates, or can be made to generate, ConfirmedEventNotifications with the Time Stamp parameter taking the BACnetDateTime form.

135.1-2013 - 8.4.8 - CHANGE_OF_LIFE_SAFETY Tests		
	Test Method	Manual

	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	If the IUT supports AE-N-I-B, these tests may be skipped. Any of the 8.4.8 or 8.5.8 tests can be used to ensure that the IUT properly generates ConfirmedEventNotification requests using the BACnetDateTime form. The specific tests that can be executed are detailed under the test cases for the specific algorithms. As long as one of the tests is executed using ConfirmedEventNotifications and the notification that is generated contains a timestamp of the BACnetDateTime form, then this test case shall be satisfied.
	Test Directives	
	Testing Hints	
	Notes & Results	

5.23.8 Generates Event Notifications with Timestamps of the Sequence Number Form

The IUT generates, or can be made to generate, ConfirmedEventNotifications with the Time Stamp parameter taking the Sequence Number form.

135.1-2013 - 8.4.8 - CHANGE_OF_LIFE_SAFETY Tests		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	If the IUT supports AE-N-I-B, these tests may be skipped. Any of the 8.4.8 or 8.5.8 tests can be used to ensure that the IUT properly generates ConfirmedEventNotification requests using the Sequence Number form. The specific tests that can be executed are detailed under the test cases for the specific algorithms. As long as one of the tests is executed using ConfirmedEventNotifications and the notification that is generated contains a timestamp of the Sequence Number form, then this test case shall be satisfied.
	Test Directives	
	Testing Hints	
	Notes & Results	

5.23.9 Mode Transition Tests when Event State is Maintained

135.1-2013 - 8.4.8.7 - Mode Transition Tests when Event State is Maintained		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	Must be executed
	Test Directives	
	Testing Hints	
	Notes & Results	

5.23.10 Supports Event_Message_Texts property

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

BTL - 7.3.1.X4 - Event_Message_Texts Tests		
	Test Method	

	Configuration	As per BTL Specified Tests.
	Test Conditionality	Must be executed.
	Test Directives	Repeat test once for each object type in the IUT that contains an Event_Message_Texts property.
	Testing Hints	
	Notes & Results	

5.23.11 Supports Event_Message_Texts_Config property

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

BTL - 7.3.1.X5 - Event_Message_Texts_Config Test		
	Test Method	Manual
	Configuration	As per BTL Specified Tests.
	Test Conditionality	Must be executed.
	Test Directives	Repeat for each supported transition type (TO_OFFNORMAL, TO_FAULT, TO_NORMAL). Different objects may be selected for different transitions.
	Testing Hints	
	Notes & Results	

[In BTL Specified Tests, add five new tests 7.3.2.15.X5 through 7.3.2.15.X9 as indicated.]

7.3.2.15.X5 Writable Tracking_Value

Purpose: This test case verifies that Present_Value equals Tracking_Value, when Tracking_Value is writable.

Test Concept: It verifies the interrelationship between the Tracking_Value, Status Flags and Present_Value properties. This test applies to Life Safety Zone and Life Safety point object. The tester will select one instance of each appropriate object type and test it as described.

Configuration Requirements: The test shall start with Event_State equal to NORMAL. If writing to the Tracking_Value is only possible while Out_Of_Service equals TRUE, then the test shall start with Out_Of_Service equal to TRUE. If the Out_Of_Service property of the object under test is not writable, and the value of the Tracking_Value property cannot be changed by other means, then this test shall be omitted.

Test Steps:

1. VERIFY Event_State = Normal
2. WRITE Tracking_Value = X (any value that corresponds to an Event_State of NORMAL)
3. VERIFY Tracking_Value = X
4. VERIFY Present_Value = X

7.3.2.15.X6 Supports Writable Mode property

Purpose: To verify that the Mode property takes one of the values found in the Accepted_Modes property.

Test Concept: It verifies the interrelationship between the Mode, and Accepted_Modes properties. This test applies to Life Safety Zone and Life Safety point object. The tester will select one instance of each appropriate object type and test it as described.

Test Steps:

1. READ AM = Accepted_Modes
2. TRANSMIT WriteProperty-Request
   'Object Identifier' = (the object being tested),
   'Property Identifier' = Mode,
   'Property Value' = X (Any valid value from list of AM)
3. RECEIVE SimpleACK-PDU
4. VERIFY Mode = X
5. TRANSMIT WriteProperty-Request
   'Object Identifier' = (the object being tested),
   'Property Identifier'= Mode,
   'Property Value' = X (Any invalid value, which is not present in AM)
6. RECEIVE BACnet-Error-PDU,
   Error Class = PROPERTY,
   Error Code = VALUE_OUT_OF_RANGE

7.3.2.15.X7 Support Operation_Expected Property

Purpose: To verify that the Operation_Expected property takes on the value of ConfirmedEventNotification-Request.

Test Concept: It verifies the interrelationship between the Operation_Expected property, and ConfirmedEventNotification-Request. This test applies to Life Safety Zone and Life Safety point object. The IUT will select one instance of each appropriate object type and test it as described.

Test Steps:

1. MAKE (the IUT send an ConfirmedEventNotification)
2. RECEIVE ConfirmedEventNotification-Request,
   'Process Identifier' = (any valid process identifier),
   'Initiating Device Identifier' = TD,
   'Event Object Identifier' = (any Life-Safety object),
   'Time Stamp' = (the current local time),
   'Notification Class' = (any valid notification class),
   'Priority' = (any valid priority),
   'Event Type' = CHANGE-OF-LIFE-SAFETY,
   'Message Text' = (any character string),
   'Notify Type' = ALARM | EVENT,
   'AckRequired' = TRUE |FALSE,
   'From State' = NORMAL,
   'To State' = (any non-normal state appropriate to the event type),
   'Event Values' = (New State: (Any Valid State), New-Mode: (Any Valid Mode), Status-Flag: (TRUE, FALSE, ?, ?), Operation_Expected: (“X “, Any Valid operation))
3. VERIFY Operation_Expected = X (operation expected in the step 2)

7.3.2.15.X8 Support Writable Member_Of property

Purpose: To verify that the Member_Of property takes only supported values of the Life Safety objects within the IUT.

Test Concept: If the property is writable and is restricted to referencing objects within the containing device, an attempt to write a reference to an object outside the containing device into this property shall cause a Result (-), if the property is not writable and if the value of the property cannot be changed by other means, then this test shall be omitted. The tester will select one instance of each appropriate object type and test it as described.

Test Steps:

```
1.  TRANSMIT WriteProperty-Request,
        Object Identifier' =        (life safety object),
        'Property Identifier' =     Member_Of
        'Property Value' =          X (any valid life safety object)
2.  RECEIVE Simple-ACK-PDU,
3.  TRANSMIT ReadProperty-Request,
        'Object Identifier' =       (life safety object),
        'Property Identifier' =     Member_Of
4.  RECEIVE ReadProperty-ACK,
        'Object Identifier' =       (the object being tested),
        'Property Identifier' =     Member_Of
        'Property Value' =          X (the value used in step 1)
```

7.3.2.15.X9 Silenced Property test

Purpose: This test verifies the behavior of Silenced property.

Test Concept: Verify the interrelationship between the Silenced property and any audible or visual indication that has been silenced by the receipt of a LifeSafetyOperation service request or a local process. If the Silenced property of the object under test is unchanging by means of a LifeSafetyOperation service requests, because none of the silencing operations are supported, then this test shall be omitted. This test applies to Life Safety Zone and Life Safety Point object. The tester will select one instance of each appropriate object type and test it as described.

Test Steps:

```
1.  READ InitialSilencedState = Silenced
2.  TRANSMIT LifeSafetyOperation-Request,
        'Requesting Process Identifier' =   (any valid identifier),
        'Requesting Source' =       (any valid character string),
        'Request' =         (any supported LifeSafetyOperation request transmitted to silence the sounder/strobe),
        'Object Identifier' =       (the selected object)
3.  RECEIVE BACnet-SimpleACK-PDU
4.  CHECK (Sounder/Strobe inactive)
5.  READ ResultingSilencedState = Silenced
6.  CHECK (the ResultingSilencedState is equal to the InitialSilencedState, modified by the LifeSafetyOperation
    request transmitted)
```

BTL-TP15.1-0.1.0 File object

A device claiming File object at Protocol_Revision 2 or higher shall comply with the following section.

Rationale: The uses of File for a purpose other than Backup and Restore, all seem to be essentially proprietary.

[In BTL Checklist, add File object type to Section 3, Objects]

Support	Listing	Option
. . .		
File		
	R	Base Requirements
	C[1]	Supports DM-BR-B
	C[1]	Supports a File object for a purpose other than Backup and Restore
	O	Contains a writable File for a purpose other than Backup and Restore
[1] At least one of these options is required if the IUT supports the File object type.		

[In BTL Test Plan, add File object tests in section 3.X48]

3.X48 File

3.X48.1 Base Requirements

For File object, there are no base requirements.

3.X48.2 Supports DM-BR-B

The IUT supports a data File that is readable and writable during Backup and Restore using AtomicReadFile and AtomicWriteFile requests.

Verify Checklist		
	Test Method	Manual
	Configuration	
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for Device Management - Backup and Restore - B in the Checklist.
	Testing Hints	
	Notes & Results	

3.X48.3 Supports a File object for a purpose other than Backup and Restore

For a device which contains a File object for a purpose other than Backup and Restore, there are no testing requirements.

3.X48.4 Contains a writable File for a purpose other than Backup and Restore

For a device which contains a writable File object for a purpose other than Backup and Restore, there are no testing requirements.

BTL-TP15.2-0.1.0: Load Control object

Devices claiming support for a Load Control object must claim support for Protocol_Revision 6 and comply with the following section.

Overview:

Addendum 135-2004*e* added the Load Control object. This document makes needed changes in the BTL Test Package to claim Load Control object.

These changes are not contained in any SSPC proposal. This testing ensures coverage for Load Control statements including:

- If no shed request is pending or active, Start_Time shall contain an unspecified datetime value.
- If a load control command has been issued, and execution of the command has completed, Start_Time shall be reset by the device to contain an unspecified datetime value.
- If no shed request is pending or active, Shed_Duration shall be zero.
- If a load control command has been issued, and execution of the command has completed, Shed_Duration shall be reset by the device to zero.
- If a shed request is received with no value written to this property, Duty_Window shall be set to some pre-agreed upon value.
- If a load control command has been issued, and execution of the command has completed, Duty_Window shall be reset by the device to this pre-agreed value.
- If a load control command has been issued, and execution of the command has completed, Requested_Shed_Level shall be reset to the default value appropriate to the choice of Requested_Shed_Level used for the last command.
- Load Control objects Requested_Shed_Level properties are required to support the LEVEL choice. Support for the PERCENT and AMOUNT choices is optional.
- Provides writability tests of Requested_Shed_Level, Start_Time, Shed_Duration, Duty_Window, Enable, and Shed_Levels.

[In BTL Checklist, add Load Control object type to Section 3, Objects]

Support	**Listing**	**Option**
Load Control Object		
	R	Base Requirements
	R	Supports writable Requested_Shed_Level to LEVEL choice
	O	Supports writable Reliability property
	O	Supports writable Requested_Shed_Level to PERCENT choice
	O	Supports writable Requested_Shed_Level to AMOUNT choice

[In BTL Test Plan, add Load Control object tests in section 3.X53. In the following addition of new clauses of the Test Plan, these are indicated as entirely new sections verbatim, with plain text, verbatim **bold**, or verbatim ***bold-italic*** as shown.]

3.X53 Load Control Object

3.X53.1 Base Requirements

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

BTL - 7.3.2.X53.2 - Shed_Levels property test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X53.2 Supports Requested_Shed_Level to LEVEL choice

The Requested_Shed_Level property in Load Control objects is writable to LEVEL choice.

BTL - 7.3.2.X53.1 - Requested_Shed_Level property test with LEVEL choice		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X53.3 Supports Writable Reliability Property

The Reliability property in Load Control objects is writable.

BTL - 7.3.2.X53.3 - Load Control Status_Flags and Reliability Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If Reliability is writable, this test must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X53.4 Supports Requested_Shed_Level to PERCENT choice

The Requested_Shed_Level property in Load Control objects is writable to PERCENT choice.

BTL - 7.3.2.X53.4 - Requested_Shed_Level property test with PERCENT choice		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If no object can be made to meet the configuration requirements, this test shall be skipped.
	Test Directives	
	Testing Hints	

	Notes & Results	

3.X53.5 Supports Requested_Shed_Level to AMOUNT choice

The Requested_Shed_Level property in Load Control objects is writable to AMOUNT choice.

BTL - 7.3.2.X53.5 - Requested_Shed_Level property test with AMOUNT choice		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If no object can be made to meet the configuration requirements, this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	

[In BTL Specified Tests, add Load Control object tests in section 3.X53. Since these are entirely new tests, these are indicated with plain text.]

7.3.2.X53 Load Control Object Tests

The Load Control object defines a standardized object whose properties represent the externally visible characteristics of a mechanism for controlling load requirements. A BACnet device can use a Load Control object to allow external control over the shedding of a load that it controls. The mechanisms by which the loads are shed are not visible to the BACnet client. The Load Control Object utilizes parameter control through its writable Requested_Shed_Level, Start_Time, Shed_Duration, Duty_Window, Enable and Shed_Levels properties.

7.3.2.X53.1 Requested_Shed_Level property test with LEVEL choice

Reason for Change: This test is not specified in any SSPC proposal.

Purpose: To verify the performance of a shed request with LEVEL choice.

Test Concept: The Requested_Shed_Level property of the Load Control object is set to a LEVEL choice and it is verified that the series of required actions which that sets into operation occur correctly.

Configuration Requirements: The IUT shall be configured so that Present_Value is equal to SHED_INACTIVE, preceding the beginning of this test. Writing Start_Time and/or Shed_Duration with values such that current time is after ST+SD forces Present_Value to become equal to SHED_INACTIVE.

Test Steps:

```
    VERIFY Requested_Shed_Level = (one of the default Requested_Shed_Level values for a previous shed
        request, not necessarily the LEVEL default of 0)
    V ERIFY Expected_Shed_Level = (that same default Requested_Shed_Level value)
    VERIFY Actual_Shed_Level = (that same default Requested_Shed_Level value)
    VERIFY Present_Value = SHED_INACTIVE
    VERIFY Shed_Duration = 0
    VERIFY Start_Time = (the fully unspecified datetime value)
    VERIFY Duty_Window = (PAV, the pre-agreed upon value)
    WRITE Enable = TRUE
    WRITE Shed_Duration = (SD, any value appropriate to the object)
    WRITE Start_Time = (ST, any value preceding the beginning of this test)
    WRITE Duty_Window = (DW, any value appropriate to the object)
```

```
WRITE Requested_Shed_Level = (a value appropriate to the object with a LEVEL choice, that is not equal to the default value: 0)
-- the above four writes can occur in any order, but it is needful that Enable becomes TRUE before the others. Each of these writes is a reconfiguration if the current time is prior to Start_Time. A reconfiguration is what forces the Present_Value to SHED_REQUEST_PENDING, so that can be observed after the first write and also is observable in-between any of the writes.
VERIFY Present_Value = SHED_REQUEST_PENDING
WAIT (until the shed request has started, typically at Start_Time but it can start earlier to achieve compliance at Start_Time)
VERIFY Present_Value = (SHED_REQUEST_PENDING or SHED_COMPLIANT or SHED_NONCOMPLIANT)
IF (current time is before ST, but the shed request has started) THEN
    VERIFY  Present_Value = SHED_NONCOMPLIANT
IF (current time is at or after ST)
    VERIFY Present_Value = (SHED_COMPLIANT or SHED_NONCOMPLIANT)
IF (current time is after ST+DW and Actual_Shed_Level does not comply with Requested_Shed_Value)
    VERIFY Present_Value = SHED_NONCOMPLIANT
VERIFY Shed_Duration = SD
VERIFY Start_Time = ST
VERIFY Duty_Window = DW
VERIFY Expected_Shed_Level = (any value appropriate to the choice, that is not equal to the default value)
VERIFY Actual_Shed_Level = (any value appropriate to the choice, that is not equal to the default value)
-- the above VERIFY statements apply all through the time that there is a pending or active shed event
WAIT (until the shed request has completed, at ST+SD)
VERIFY Requested_Shed_Level = 0
VERIFY Expected_Shed_Level = (0, that same Default LEVEL value)
VERIFY Actual_Shed_Level = (0, that same Default LEVEL value)
VERIFY Shed_Duration = 0
VERIFY Start_Time = (the fully unspecified datetime value)
VERIFY Duty_Window = PAV
```

Notes to Tester: The writing of Duty_Window can be skipped, for the tester to see that the VERIFY Duty_Window = DW during a pending or active shed event, that property takes on PAV, the pre-agreed upon value.

7.3.2.X53.2 Shed_Levels property test

Reason for Change: This test is not specified in any SSPC proposal.

Purpose: To verify writability of Shed_Levels property and verify that when commanded with the LEVEL choice, the Load Control object shall take a shedding action described by the corresponding element in the Shed_Level_Descriptions array.

Test Concept: The Shed_Levels property of the Load Control object being tested is written to BACnetARRAY of unsigned integers representing the shed levels for the LEVEL choice of BACnetShedLevel that have meaning for this particular Load Control object. Verify that is updating correctly. The array shall be ordered by increasing shed amount.

Test Steps:

1. READ N1 = Shed_Levels, ARRAY_INDEX = 0
2. VERIFY (Shed_Level_Descriptions = N1, ARRAY_INDEX = 0)
3. WRITE Shed_Levels = (any content that is different from the current value, but nonetheless still ordered by increasing shed amount)
4. READ N2 = Shed_Levels, ARRAY_INDEX = 0 -- obtaining the length of the new value
5. VERIFY (Shed_Level_Descriptions = N2, ARRAY INDEX = 0)

7.3.2.X53.3 Load Control Status_Flags and Reliability Test

Purpose: To ensure Status_Flags reflects the Reliability property value.

Test Concept: Write to Reliability and verify the interrelationship between the Status_Flags and Reliability.

Configuration Requirements: The selected object is configured such that its Reliability is NO_FAULT_DETECTED before execution of this test. If the Reliability property is not present or not writable, then this test shall be skipped.

Test Steps:

```
1. VERIFY Reliability = NO_FAULT_DETECTED
2. VERIFY Status_Flags = (?, FALSE, ?, FALSE)
   REPEAT X = (all values of the Reliability enumeration appropriate to the object type except
                   NO_FAULT_DETECTED) DO {
       WRITE Reliability = X
       VERIFY Reliability = X
       VERIFY Status_Flags = (TRUE, TRUE, ?, FALSE)
       WRITE Reliability = NO_FAULT_DETECTED
       VERIFY Reliability = NO_FAULT_DETECTED
       VERIFY Status_Flags = (? FALSE, ?, FALSE)
   }
```

7.3.2.X53.4 Requested_Shed_Level property test with PERCENT choice

Reason for Change: This test is not specified in any SSPC proposal.

Purpose: To verify the performance of a shed request with PERCENT choice.

Test Concept: The Requested_Shed_Level property of the Load Control object is set to a PERCENT choice and it is verified that the series of required actions which that sets into operation occur correctly.

Test Steps: The test steps defined in test **7.3.2.X53.1** shall be followed except that the Requested_Shed_Level property of the Load Control object is written to a PERCENT choice, and the default value for a shed request with PERCENT choice in Requested_Shed_Level, Expected_Shed_Level, and Actual_Shed_Level properties is 100

7.3.2.X53.5 Requested_Shed_Level property test with AMOUNT choice

Reason for Change: This test is not specified in any SSPC proposal.

Purpose: To verify the performance of a shed request with AMOUNT choice.

Test Concept: The Requested_Shed_Level property of the Load Control object is set to an AMOUNT choice and it is verified that the series of required actions which that sets into operation occur correctly.

Test Steps: The test steps defined in test **7.3.2.X53.1** shall be followed except that the Requested_Shed_Level property of the Load Control object is written to an AMOUNT choice, and the default value for a shed request with AMOUNT choice in Requested_Shed_Level, Expected_Shed_Level, and Actual_Shed_Level properties is 0.0

BTL-TP15.2-1.1.0: Access Door object

Devices claiming support for an Access Door object must claim support for Protocol_Revision 6 and comply with the following section.

Overview:

Addendum 135-2004d added the Access Door object. This document makes needed changes in the BTL Test Package to claim the Access Door object.

These changes are not contained in any SSPC proposal. This testing ensures coverage for Access Door requirements.

[In BTL Checklist, add Access Door object type to Section 3, Objects]

Support	**Listing**	**Option**
Access Door Object		
	R	Base Requirements
	R	Supports Command Prioritization
	S	Supports writable Out_Of_Service properties
	C[1]	Supports Door_Status
	O	Supports Lock_Status
	O	Supports Secured_Status
	O	Supports Door_Unlock_Delay_Time
	O	Supports Masked_Alarm_Values
	O	Supports Intrinsic Reporting
	O	Contains an object with Reliability_Evaluation_Inhibit Property
[1] If Secured_Status is supported, this is required.		

Changes:

[In BTL Test Plan, add Access Door object tests in section 3.X55]

3.X55 Access Door Object

3.X55.1 Base Requirements

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

BTL - 7.3.2.X55.1.X1 – Commandable Present_Value Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

	Notes & Results	

3.X55.2 Supports Command Prioritization

135.1-2013 - 7.3.1.2 - Relinquish Default Test		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	If no object can be made to meet the configuration requirements, this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	
135.1-2013 - 7.3.1.3 - Command Prioritization Test		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X55.3 Supports Writable Out_Of_Service Properties

The IUT contains or can be made to contain writable Out_Of_Service property.

135.1-2013 - 7.3.1.1 - Out_Of_Service, Status_Flags, and Reliability Tests		
	Test Method	Manual
	Configuration	The test shall be executed using an Access Door object
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.2.X55.1.X2 - Door_Status, Lock_Status and Door_Alarm_State Tests		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If neither Door_Status, Lock_Status or Door_Alarm_State is supported, this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X55.4 Supports Door_Status

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

BTL - 7.3.2.X55.1.X3 - Door_Status with physical door status Tests		
	Test Method	Manual
	Configuration	As per ***BTL Specified 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	
	Notes & Results	

3.X55.5 Supports Lock_Status

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

BTL - 7.3.2.X55.1.X4 – Lock_Status Tests		
	Test Method	Manual
	Configuration	As per ***BTL Specified 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	
	Notes & Results	

3.X55.6 Supports Secured_Status

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

BTL - 7.3.2.X55.1.X5 – Secured_Status Tests		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If the Secured_Status property is permanently configured to have the value UNKNOWN then this test shall be omitted.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X55.7 Supports Door_Unlock_Delay_Time

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

BTL - 7.3.2.X55.1.X6 - Door_Unlock_Delay_Time Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.

	Test Directives	
	Testing Hints	
	Notes & Results	

3.X55.8 Supports Masked_Alarm_Values

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

BTL - 7.3.2.X55.1.X7- Masked_Alarm_Values Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If Out_Of_Service is not writeable and cannot be set to TRUE by any other means, this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X55.9 Supports Intrinsic Reporting

The IUT supports intrinsic reporting.

BTL - 7.3.2.X55.1.X8- Door_Open_Too_Long Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

3.X55.10 Contains an object with Reliability_Evaluation_Inhibit Property

The IUT contains or can be made to contain a Reliability_Evaluation_Inhibit property that is configurable to a value of TRUE.

BTL - 7.3.1.X8.1 - Reliability_Evaluation_Inhibit Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If no object exists in the IUT for which fault conditions can be generated then this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	
BTL - 7.3.1.X8.2 - Reliability_Evaluation_Inhibit Summarization Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.

	Test Conditionality	If no object exists in the IUT for which fault conditions can be generated then this test shall be skipped.
	Test Directives	
	Testing Hints	
	Notes & Results	

Changes:

[In BTL Specified Tests, add Access Door object specific tests in section 7.3.2.X]

7.3.2.X55.1.X1 Commandable Present_Value Test

Purpose: To verify that writing to the Present_Value will cause a corresponding change to the physical output.

Test Concept: The IUT shall be configured with a door control output that can be observed during the test. The Present_Value property is written with each of the following values: UNLOCK, LOCK, PULSE_UNLOCK, EXTENDED_PULSE_UNLOCK and the Access Door object is monitored to ensure that the door locks and unlocks appropriately.

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.

Test Steps:

```
-- Test UNLOCK value
1.  WRITE Present_Value = UNLOCK
2.  WAIT (Internal Processing Fail Time)
3.  IF (Lock_Status is present) THEN

        VERIFY Lock_Status = UNLOCKED
4.  CHECK (that the door control output is in a state that would cause the door to be unlocked)

-- Test LOCK value
5.  WRITE Present_Value = LOCK
6.  WAIT (Internal Processing Fail Time)
7.  IF (Lock_Status is present) THEN
        VERIFY Lock_Status =     LOCKED
8.  CHECK (that the door control output  is in a state that would cause the door to be locked)

-- Test PULSE_UNLOCK value
9.  WRITE Present_Value = PULSE_UNLOCK
10. WAIT (Internal Processing Fail Time + Door_Unlock_Delay_Time if present)
11. IF (Lock_Status is present) THEN
        VERIFY Lock_Status = UNLOCKED
12. CHECK (that the IUT is in a state that would cause the door to be unlocked)
13. WAIT (Door_Pulse_Time)
14. VERIFY Present_Value = LOCK
15. IF (Lock_Status is present) THEN
        VERIFY Lock_Status = LOCKED
16. CHECK (that the door control output  is in a state that would cause the door to be locked)

-- Test EXTENDED_PULSE_UNLOCK value
17. WRITE Present_Value = EXTENDED_PULSE_UNLOCK
18. WAIT (Internal Processing Fail Time + Door_Unlock_Delay_Time if present)
```

7.3.2.X55.1.X2 Door_Status, Lock_Status and Door_Alarm_State Tests

Purpose: This test case verifies that Door_Status, Lock_Status and Door_Alarm_State properties are writable when Out_Of_Service is TRUE.

Test Concept: Set Out_Of_Service to TRUE and then make sure one at a time that Door_Status, Lock_Status and Door_Alarm_State, if present, are writable.

Configuration Requirements : If the Out_Of_Service property of this object is not writable, and if the Out_Of_Service property cannot be changed by other means, then this test shall be omitted. All writes to the Present_Value shall be performed at a priority higher (numerically smaller) than any internal algorithms writing to this property. For testing Door_Alarm_State, test only values listed in either the Alarm_Values or Fault_Values.

Test Steps:

```
1.  MAKE (Out_Of_Service TRUE)
2.  VERIFY Status_Flags = (?,?,?,TRUE)
3.  IF (Door_Status is present) THEN
        REPEAT X = (all values of the Door_Status enumeration values supported by the property)
            DO {
                WRITE Door_Status = X
                VERIFY Door_Status = X
            }
4.  IF (Lock_Status is present) THEN
        REPEAT X = (all values of the Lock_Status enumeration values supported by the property)
            DO {
                WRITE Lock_Status = X
                VERIFY Lock_Status = X
            }
5.  IF (Door_Alarm_State is present) THEN
        REPEAT X = (all values of the Door_Alarm_State enumeration values supported by the property)
            DO {
                WRITE Door_Alarm_State = X
                VERIFY Door_Alarm_State = X
            }
```

7.3.2.X55.1.X3 Door_Status with physical door status Tests

Purpose: To verify that the Door_Status property reflects the state of the physical door (CLOSED, OPENED, UNUSED and DOOR_FAULT if the object supports detecting door faults).

Test Concept: The IUT is configured to monitor the state of a physical door. The physical door may be represented by a BACnet input object or through some proprietary method.

Configuration Requirements: The IUT shall be configured such that it can determine the state of a door. The Access_Door object associated with this physical door shall be configured with Out_Of_Service = FALSE.

Test Steps:

```
1.  MAKE (set physical door to the closed state)
2.  VERIFY Door_Status = CLOSED
3.  MAKE (set physical door to the opened state)
4.  VERIFY Door_Status = OPENED
5.  IF (the object supports detecting door faults)
```

```
        MAKE (set the physical door to a state that would cause the Door_Status to take on a value of
                DOOR_FAULT)
        VERIFY Door_Status = DOOR_FAULT
6.   IF (possible to remove a door status input associated with the door)
        MAKE (remove a door status input associated with the door)
7.      VERIFY Door_Status = UNUSED | UNKNOWN
```

7.3.2.X55.1.X4 – Lock_Status Tests

Purpose: To verify that the Lock_Status property reflects the state of the physical lock. (LOCKED, UNLOCKED and LOCK _FAULT if the object supports detecting lock faults).

Test Concept: The IUT monitors the state of a physical lock. The state of the physical lock may be represented by a BACnet input object or through some proprietary method.

Configuration Requirements: The IUT shall be configured such that it can monitor the state of the physical lock. The Access_Door object associated with this physical door shall be configured with Out_Of_Service = FALSE. The physical lock shall be manipulated other than through the Access Door object.

Note to tester: The physical lock shall be manipulated other than through the Access Door object.

Test Steps:

```
1.  MAKE (set the physical lock to a state that would cause the Lock_Status to take on a value of
            LOCKED)
2.  VERIFY Lock_Status = LOCKED
3.  MAKE (set the physical lock to a state that would cause the Lock_Status to take on a value of
            UNLOCKED)
4.  VERIFY Lock_Status = UNLOCKED
5.  IF (the object and the lock support detecting lock faults)
        MAKE (set the physical lock to a state that would cause the Lock_Status to take on a value of
                LOCK_FAULT)
        VERIFY Lock_Status = LOCK_FAULT
```

7.3.2.X55.1.X5 – Secured_Status Tests

Purpose: To verify that the Secured_Status property reflects the state of the physical lock, the physical door and the state of the Access Door object.

Test Concept: Start the test by creating a condition where the Secured_Status = SECURED. Then create various conditions one at a time to verify that the Secured_Status becomes UNSECURED when it should.

Configuration Requirements: All writes to the Present_Value shall be performed at a priority higher than any internal algorithms writing to this property. If this object supports intrinsic reporting then the Alarm_Values property shall be empty. If this object supports the Masked_Alarm_Values property then it shall be empty. Out_Of_Service is FALSE.

Test Steps:

```
-- Create a condition where the Secured_Status becomes SECURED
1.  WRITE Present_Value = LOCK
2.  WAIT (Internal Processing Fail Time)
3.  VERIFY Status_Flags = (FALSE ?,?, ?)
4.  IF (Lock_Status property is present)
        MAKE (Lock_Status = LOCKED or UNUSED)
5.  MAKE (Door_Status =  CLOSED or UNUSED)
```

```
-- Verify that the Secured_Status is SECURED when it should
6.  VERIFY Secured_Status = SECURED

-- Verify that Secured_Status is UNSECURED when Present_Value is anything other than LOCKED
7.  REPEAT X = (UNLOCK, PULSE_UNLOCK, EXTENDED_PULSE_UNLOCK) DO {
        WRITE Present_Value = X
        WAIT (Internal Processing Fail Time)
        VERIFY Secured_Status = UNSECURED
    }
-- Recreate a condition where the Secured_Status becomes SECURED again
8.  WRITE Present_Value = LOCK
9.  WAIT (Internal Processing Fail Time)
10. VERIFY Secured_Status = SECURED

-- Verify that Secured_Status is UNSECURED when Masked_Alarm_Value, if exist, is NOT empty
11. IF (Masked_Alarm_Values is present) THEN
        MAKE (Masked_Alarm_Values = (any valid BACnetDoorAlarmState enumeration))
        WAIT(Internal Processing Fail Time)
        VERIFY Secured_Status = UNSECURED

-- Recreate a condition where the Secured_Status becomes SECURED again
        MAKE ( Masked_Alarm_Values = {})
        WAIT (Internal Processing Fail Time)
        VERIFYSecured_Status = SECURED

-- Verify that Secured_Status is UNSECURED when Lock_Status, if present, is anything other than LOCKED or UNUSED
12. IF (Lock_Status property is present) THEN
        REPEAT X = (UNLOCKED. UNKNOWN, LOCK_FAULT) DO {
            MAKE (Lock_Status = X)
            WAIT (Internal Processing Fail Time)
            VERIFY Secured_Status = UNSECURED
        }
        REPEAT X = (LOCKED, UNUSED) DO {
            MAKE (Lock_Status = X)
            VERIFY Secured_Status = SECURED
        }

-- Verify that Secured_Status is UNSECURED when Door_Status, is anything other than CLOSED or UNUSED
13. REPEAT X = (OPEN, UNKNOWN, DOOR_FAULT) DO {
        MAKE (Door_Status = X)
        WAIT (Internal Processing Fail Time)
        VERIFY Secured_Status = UNSECURED
    }
    REPEAT X = (CLOSED, UNUSED) DO {
        MAKE (Door_Status = X)
        WAIT (Internal Processing Fail Time)
        VERIFY Secured_Status = SECURED
    }

-- Verify that Secured_Status is UNSECURED when In_Alarm bit of Status_Flag is True
14. IF (Alarming is supported) THEN
        IF (Alarm_Values is writable) THEN
            WRITE Alarm_Values = { AV: any valid value}
        MAKE (trigger an alarm by using a physical door/lock to create the door alarm state AV)
        WAIT (Internal Processing Fail Time + Time_Delay)
        VERIFY Status_Flags = (TRUE, FALSE, ?, ?)
        VERIFY Secured_Status = UNSECURED
```

7.3.2.X55.1.X6 Door_Unlock_Delay_Time Test

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

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

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

Test Steps:

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

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

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

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

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

7.3.2.X55.1.X7 Masked_Alarm_Values Tests
Purpose: To verify that the Masked_Alarm_Values prevents an intrinsic alarm from occurring.

Test Concept: The Access Door is verified to be in an Out_Of_Service stateand is not in an alarm state. Then a non-NORMAL enumeration value of BACnetDoorAlarmState X is written to the Door_Alarm_State and the Access Door object transitions to an alarm state. X is written to the Masked_Alarm_Value and Door_Alarm_State is checked to verify it returned to NORMAL. The sequence is repeated for all non-NORMAL enumeration values of BACnetDoorAlarmState.

Configuration Requirements: The Masked_Alarm_Values list shall be empty at the start of this test. Out_Of_Service shall be set to TRUE to allow writing to the Door_Alarm_State property. If Out_Of_Service is not writeable and cannot be set to TRUE by any other means, this test shall be skipped. The enumeration BACnetDoorAlarmState value X to be used in the test has to be present in either the Alarm_Values or Fault_Values property.

Test Steps:

```
1. VERIFY Status_Flags = (FALSE ?, ?, TRUE)
2. VERIFY Door_Alarm_State = NORMAL
3. REPEAT X = (all valid values of the enumeration BACnetDoorAlarmState except NORMAL)
       DO {
               WRITE Door_Alarm_State = X
               WAIT (Internal Processing Fail Time)
               VERIFY Status_Flags = (TRUE ?, ?, TRUE)
               WRITE Masked_Alarm_Values= { X }
               WAIT (Internal Processing Fail Time)
               VERIFY Door_Alarm_State = NORMAL
               VERIFY Status_Flags = (FALSE ?, ?, TRUE)
               WRITE Masked_Alarm_Values= { }
               WAIT (Internal Processing Fail Time)
       }
```

7.3.2.X55.1.X8 Door_Open_Too_Long Test
Purpose: To verify that the DOOR_OPEN_TOO_LONG condition is generated when the Access Door object is commanded to the LOCK state but the physical door remains open beyond Door_Open_Too_Long_Time.

Test Concept: Setup the Access Door object to trigger alarm on DOOR_OPEN_TOO_LONG state using Alarm_Values and Masked_Alarm_Values. Next, set the physical door to the closed state to confirm that the Access Door object is in NORMAL state. Then, unlock the physical door and set the physical door to the open state. Finally, command the Access Door object to LOCK and verify that the Door_Alarm_State changes to DOOR_OPEN_TOO_LONG after the specified Time_Delay.

Configuration Requirements: This test shall be skipped if the IUT does not support intrinsic alarming. The IUT shall be configured such that it can determine and change the open/closed state of a door. All writes to the Present_Value are at a priority higher than any internal algorithms writing to this property. The Door_Alarm_State shall have the value NORMAL at the start of the test. The Access Door object is configured with DOOR_OPEN_TOO_LONG in the Alarm_Values property and excluded from Masked_Alarm_Values property if present.

Test Steps:

```
1.  MAKE (set the physical door to the closed state)
2.  VERIFY Door_Alarm_State = NORMAL
3.  WRITE Present_Value = UNLOCK
4.  MAKE (set the physical door to the open state)
5.  WRITE Present_Value = LOCK
6.  WAIT (Internal Processing Fail Time)
7.  WHILE (Door_Open_Too_Long_Time has not expired) DO {
        VERIFY Door_Alarm_State = NORMAL
    }
    WAIT (Time_Delay)
8.  VERIFY Door_Alarm_State = DOOR_OPEN_TOO_LONG
```