BACnet® TESTING LABORATORIES

INTERIM TEST SPECIFICATION

To Be Used with Test Package 15
Version 6
April 6, 2018

Approved by the BTL Working Group on 3/25/2018
Approved by the BTL Working Group Voting Members on 4/20/18
Published on 5/1/18.

Forward

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.

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
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-TP15.0-3.1.0 NM-CE-A Test Considerations
BTL-TP15.0-4.1.0 Read-only Recipient_List Test Considerations (Protocol_Revision 13 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 15.0 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.

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	This test shall be applied to a Network Port object.
Testing Hints	
Notes & Results	

[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; $1 \leq N \leq$ 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; $1 \leq N \leq$ 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; $1 \leq N \leq$ 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; $1 \leq N \leq$ number of doors of a car) and 'Property Value' = (any valid value X; $X \leq$ 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; $1 \leq N \leq$ 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; $1 \leq N \leq$ 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.

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.X41.2 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 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.X41.5 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.X41.6 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.X41.7 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-TP15.0-3.1.0 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
   })