BACnet® TESTING LABORATORIES

INTERIM TEST SPECIFICATION

To Be Used with Test Package 16.1
Version 10
May 5, 2020

Approved by the BTL Working Group on April 30, 2020.
Approved by the BTL Working Group Voting Members on May 18, 2020.
Published on May 20, 2020.

Foreward

The purpose of this document is to define interim tests and other test package changes made to support testing of a device that supports functionality currently not covered in the released BTL Test Package. This document shall be applied and used with BTL Test Package 16.1.

Vendors who are planning to submit a device for testing and who implement Protocol_Revision 17 and higher, or which contain functionality not covered by the Official Test Package, should use this Interim Test document.

Please note that if the device contains functionality not yet covered by the official Test Package, nor by the Interim Tests document, development of new tests may 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.

In the following document, language to be added to existing clauses of ANSI/ASHRAE 135.1-2013 or any part of the Test Package 16.0 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 Checklist and BTL Test Plan Changes

This section of the document contains interim changes to the BTL Checklist and the BTL Test Plan documents to support testing of products with functionality outside the scope of the official test plan.

This section is ordered the same as the BTL Checklist and BTL Test Plan documents to allow easy navigation of the material.

All test changes can be found in the next major section.

1.2 Testing Virtual Network Gateways

The BTL Test Package does not provide adequate direction on testing of virtual network gateways. These changes direct the tester to develop 2 separate Checklists, one for the functionality in the virtual router, and another for the superset of functionality that I supported in virtual devices.

Checklist Changes

[Modify clause 1 in the BTL Functionality Checklist]

1 Introduction

The ***BTL Functionality Checklist*** identifies the testable options implemented by the IUT. The table is divided out into sections by functionality. In general, each section maps onto a BIBB, object type, or functional category. Each section has a Base Requirements option and if the BIBB, object type or functional category is supported by the IUT, this item must be selected. In addition, any other option in the section that has a Listing Code of R or BTL-R must be selected.

There are some items in the table that are already marked with an X in the 'Support' column. These are items that all BACnet devices must implement.

The Listing column indicates whether the option is required or not. The codes in the table are:

R = Required. Items marked with this listing code are required for a listing if the IUT implements the associated BIBB, object type, or functional category.

BTL-R = Required by BTL. Items marked with this listing code are required for a listing if the IUT implements the associated BIBB, object type, or functional category.

C = Conditionally Required. Items marked with this listing code may be required for a listing if the IUT implements the associated BIBB, object type, or functional category. The conditions under which the item will be required are identified in a footnote in the Checklist table.

BTL-C = Conditionally Required by BTL. Items marked with this listing code may be required for a listing if the IUT implements the associated BIBB, object type, or functional category. The conditions under which the item will be required are identified in a footnote in the Checklist table.

S = Suggested. The BTL suggests that all IUTs implement this option if they implement the associated BIBB, object type, or functional category.

O = Optional. Items marked with this listing code are optional.

N = Not recommended. The BTL recommends against IUTs implementing this option due to possible interoperability or performance problems related with the option.

The 'Option' column names the functional item. For each item there is a corresponding item of the same name in the ***BTL Test Plan***. The corresponding item in the ***BTL Test Plan*** provides a more detailed description of the option.

Once filled out, this document will be used to identify the tests to apply to the IUT. By relating the selected items in this table to items in the ***BTL Test Plan***, the tester will have a list of all tests that must be applied to the IUT.

If the IUT supports GW-VN-B, then a separate BTL Checklist shall be filled out describing the functionality of the virtual devices. The virtual device checklist shall document all of the functionality that is supported in the virtual devices even if it cannot all be supported in a single virtual device.

Test Plan Changes

[Add section 1.2 into BTL Test Plan]

1.2 Testing Virtual Network Gateways

This test plan was developed to test a single BACnet device but BACnet virtual gateways are different from other BACnet devices in that they represent 2 or more devices: the virtual router and one or more virtual devices. The functionality of the virtual router device might be very different than the functionality of the virtual devices and thus warrants separate testing.

The Test Plan shall be applied to the virtual router based on its BTL Checklist and on a virtual device based on its BTL Checklist.

[Add into GW-VN-B Base Requirements]

Verify Virtual Devices		
	Test Conditionality	Must be executed
	Test Directives	Test the virtual devices as per their BTL Checklist.
	Testing Hints	Similar to testing derivative products, the functionality supported might need to be spread over 2 or more virtual devices. In such cases, to which of the virtual devices any particular test is applied is left up to the test as long as all applicable tests are executed.

3.2 Analog Output Object

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, add entry into section 3.2.2 Supports Command Prioritization]

BTL - 7.3.1.X1 - Current_Command_Priority Tracking Test		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision 17 or higher.
	Test Directives	
	Testing Hints	

3.3 Analog Value Object

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, add entry into section 3.3.3 Supports Command Prioritization]

BTL - 7.3.1.X1 - Current_Command_Priority Tracking Test		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision 17 or higher.
	Test Directives	
	Testing Hints	

3.6 Binary Output Object

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, add entry into section 3.6.2 Supports Command Prioritization]

BTL - 7.3.1.X1 - Current_Command_Priority Tracking Test		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision 17 or higher.
	Test Directives	
	Testing Hints	

3.7 Binary Value Object

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, add entry into section 3.7.5 Supports Command Prioritization]

BTL - 7.3.1.X1 - Current_Command_Priority Tracking Test		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision 17 or higher.
	Test Directives	
	Testing Hints	

3.15 Multi-state Output Object

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, add entry into section 3.15.2 Supports Command Prioritization]

BTL - 7.3.1.X1 - Current_Command_Priority Tracking Test		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision 17 or higher.
	Test Directives	
	Testing Hints	

3.16 Multi-state Value Object

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, add entry into section 3.16.4 Supports Command Prioritization]

BTL - 7.3.1.X1 - Current_Command_Priority Tracking Test		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision 17 or higher.
	Test Directives	
	Testing Hints	

3.24 Bitstring Value Object

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, add entry into section 3.24.3 Supports Command Prioritization]

BTL - 7.3.1.X1 - Current_Command_Priority Tracking Test		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision 17 or higher.
	Test Directives	
	Testing Hints	

3.25 CharacterString Value Object

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, add entry into section 3.25.3 Supports Command Prioritization]

BTL - 7.3.1.X1 - Current_Command_Priority Tracking Test		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision 17 or higher.
	Test Directives	
	Testing Hints	

3.26 Date Pattern Value Object

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, add entry into section 3.26.3 Supports Command Prioritization]

BTL - 7.3.1.X1 - Current_Command_Priority Tracking Test		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision 17 or higher.
	Test Directives	
	Testing Hints	

3.27 Date Value Object

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, add entry into section 3.27.3 Supports Command Prioritization]

BTL - 7.3.1.X1 - Current_Command_Priority Tracking Test		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision 17 or higher.
	Test Directives	
	Testing Hints	

3.28 DateTime Pattern Value Object

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, add entry into section 3.28.3 Supports Command Prioritization]

BTL - 7.3.1.X1 - Current_Command_Priority Tracking Test		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision 17 or higher.
	Test Directives	
	Testing Hints	

3.29 DateTime Value Object

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, add entry into section 3.29.3 Supports Command Prioritization]

BTL - 7.3.1.X1 - Current_Command_Priority Tracking Test		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision 17 or higher.
	Test Directives	
	Testing Hints	

3.30 Integer Value Object

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, add entry into section 3.30.3 Supports Command Prioritization]

BTL - 7.3.1.X1 - Current_Command_Priority Tracking Test		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision 17 or higher.
	Test Directives	
	Testing Hints	

3.31 Large Analog Value Object

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, add entry into section 3.31.3 Supports Command Prioritization]

BTL - 7.3.1.X1 - Current_Command_Priority Tracking Test		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision 17 or higher.
	Test Directives	
	Testing Hints	

3.32 OctetString Value Object

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, add entry into section 3.32.3 Supports Command Prioritization]

BTL - 7.3.1.X1 - Current_Command_Priority Tracking Test		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision 17 or higher.
	Test Directives	
	Testing Hints	

3.33 Positive Integer Value Object

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, add entry into section 3.33.3 Supports Command Prioritization]

BTL - 7.3.1.X1 - Current_Command_Priority Tracking Test		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision 17 or higher.
	Test Directives	
	Testing Hints	

3.34 Time Pattern Value Object

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, add entry into section 3.34.3 Supports Command Prioritization]

BTL - 7.3.1.X1 - Current_Command_Priority Tracking Test		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision 17 or higher.
	Test Directives	
	Testing Hints	

3.35 Time Value Object

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, add entry into section 3.35.3 Supports Command Prioritization]

BTL - 7.3.1.X1 - Current_Command_Priority Tracking Test		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision 17 or higher.
	Test Directives	
	Testing Hints	

3.40 Access Door Object

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, add entry into section 3.40.2 Supports Command Prioritization]

BTL - 7.3.1.X1 - Current_Command_Priority Tracking Test		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision 17 or higher.
	Test Directives	
	Testing Hints	

3.54 Lighting Output Object

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, add entry into section 3.54.2 Supports Command Prioritization]

BTL - 7.3.1.X1 - Current_Command_Priority Tracking Test		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision 17 or higher.
	Test Directives	
	Testing Hints	

3.55 Binary Lighting Output Object

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, add entry into section 3.55.2 Supports Command Prioritization]

BTL - 7.3.1.X1 - Current_Command_Priority Tracking Test		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision 17 or higher.
	Test Directives	
	Testing Hints	

3.56 Network Port Object

See Test Package addendum 16.1ai for interim Network Port object testing.

3.58 Elevator Group object

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

Checklist Changes

[In BTL Checklist, replace Elevator Group Object section]

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

Test Plan Changes

[In BTL Test Plan, replace section 3.58 Elevator Group Object]

3.58 Elevator Group Object

3.58.1 Base Requirements

The object contains Machine_Room_ID Property.

BTL - 7.3.2.X45.1 - Machine_Room_ID property linking with the Positive_Integer_Value Object		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

3.58.2 Supports Group_Members Property

The object contains a Group_Members Property.

BTL - 7.3.2.X45.2 - Linking of Lift Objects under Group_Members property of the Elevator Group Object		
	Test Conditionality	Must be executed if IUT supports Lift object.
	Test Directives	
	Testing Hints	
BTL - 7.3.2.X45.3 - Linking of Escalator Objects under Group_Members property of the Elevator Group Object		
	Test Conditionality	Must be executed if IUT supports Escalator object.
	Test Directives	
	Testing Hints	

3.58.3 Supports Landing_Call_Control Property

The object contains a Landing_Call_Control Property.

BTL - 7.3.2.X45.4 - Linking of Landing_Call_Control Property Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

3.59 Lift Object

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

Checklist Changes

[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
	O	Supports FAULT-to-FAULT transitions in FAULT_LISTED

Test Plan Changes

[In BTL Test Plan, replace section 3.59 Lift Object]

3.59 Lift Object

3.59.1 Base Requirements

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

BTL - 7.3.2.X47.1 - Elevator_Group property of Lift Object linking with Group_Members property of Elevator Group Object.		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

3.59.2 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 Conditionality	If this property is writable, this test must be executed.
	Test Directives	
	Testing Hints	
BTL - 7.3.2.X47.2 - Car_Moving_Direction and Car_Assigned_Direction Tracking Test		
	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	
BTL - 7.3.2.X47.3 - Car_Door_Status and Landing_Door_Status Tracking Test		
	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	
BTL - 7.3.2.X47.4 - Car_Position and Next_Stopping_Floor Tracking Test		
	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	
BTL - 7.3.2.X47.5 - Passenger_Alarm and Fault_Signals Tracking Test		
	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	
BTL - 7.3.2.X47.6 - Making_Car_Call, Car_Mode & Car_Door_Command Tracking Test		
	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	
BTL - 7.3.2.X47.7 - Assigned_Landing_Call and Registered_Car_Call Tracking Test		
	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	
BTL - 7.3.2.X47.8 - Car_Door_Zone and Car_Load Tracking Test		
	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	
BTL - 7.3.2.X47.9 - Energy_Meter and Car_Drive_Status Tracking Test		

	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	

3.59.3 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.X47.10 - Making_Car_Call and Registered_Car_Call Tests		
	Test Conditionality	This test must be executed if Making_Car_Call and Registered_Car_Call properties are present.
	Test Directives	
	Testing Hints	

3.59.4 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.X47.11 - Array Size of the Lift Object properties based on car door size		
	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	

3.59.5 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.X47.12 - Landing_Door_Status Tracks Car_Door_Status Test		
	Test Conditionality	This test must be executed if Landing_Door_Status property is present.
	Test Directives	
	Testing Hints	

3.59.6 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.X47.13 - Highest Universal floor number linking to Car_Position and Next_Stopping_Floor properties		
	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	

3.59.7 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.X47.14 Highest Universal floor number linking to Assigned_Landing_Calls, Making_Car_Call and Registered_Car_Call properties		
	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	

3.59.8 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.X47.15 Energy_Meter_Ref Property Tests		
	Test Conditionality	This test must be executed if Energy_Meter_Ref and Energy_Meter property is present
	Test Directives	
	Testing Hints	

3.59.9 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.X47.16 Higher_Deck and Lower_Deck Tests		
	Test Conditionality	This test must be executed if Higher_Deck and Lower_Deck properties are present
	Test Directives	
	Testing Hints	

3.59.10 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 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	
BTL - 7.3.1.X8.2 - Reliability_Evaluation_Inhibit Summarization Test		
	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	

3.59.11 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.X9.13 CHANGE_OF_RELIABILITY with FAULT_LISTED Algorithm (ConfirmedEventNotification)		
	Test Conditionality	This test must be executed
	Test Directives	
	Testing Hints	
BTL - 8.5.X9.14 CHANGE_OF_RELIABILITY with FAULT_LISTED Algorithm (UnconfirmedEventNotification)		
	Test Conditionality	This test must be executed
	Test Directives	
	Testing Hints	

3.59.12 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.X46.8 CHANGE_OF_STATE for Passenger_Alarm (ConfirmedEventNotification)		
	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	
BTL - 7.3.2.X46.9 CHANGE_OF_STATE for Passenger_Alarm (UnconfirmedEventNotification)		
	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	

3.59.13 Supports writable Assigned_Landing_Calls Property

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

BTL - 7.3.2.X47.17 - Linking of Assigned_Landing_Calls property of Lift Object to Landing_Calls property of Elevator Group		
	Test Conditionality	This test must be executed if Assigned_Landing_Calls is writable.
	Test Directives	
	Testing Hints	

3.59.14 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 Lift objects.

BTL - 8.5.X9.15 - CHANGE_OF_RELIABILITY FAULT-to-FAULT transitions in FAULT_LISTED		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

3.60 Escalator Object

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

Checklist Changes

[In BTL Checklist, replace Escalator Object section]

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
	O	Supports Reliability Evaluation
	O	Supports FAULT-to-FAULT transitions in FAULT_LISTED

Test Plan Changes

[In BTL Test Plan, replace section 3.60 Escalator Object]

3.60 Escalator Object

3.60.1 Base Requirements

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

BTL - 7.3.2.X46.1 Elevator_Group property of Escalator Object linking with Group_Members property of Elevator Group Object		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

3.60.2 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 Conditionality	If this property is writable, this test must be executed.
	Test Directives	
	Testing Hints	
BTL - 7.3.2.X46.2 - Energy_Meter, Power_Mode and Operation_Direction Tracking Test		
	Test Conditionality	This test must be executed if Energy_Meter or Power_Mode properties are present.

	Test Directives	
	Testing Hints	
BTL - 7.3.2.X46.3 - Passenger_Alarm and Fault_Signals Tracking Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 7.3.2.X46.4 - Escalator_Mode Tracking Test		
	Test Conditionality	This test must be executed if Escalator_Mode property is present.
	Test Directives	
	Testing Hints	

3.60.3 Supports Escalator_Mode Property

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

BTL - 7.3.2.X46.5 - Operation_Direction Tracks Escalator_Mode Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

3.60.4 Supports Energy_Meter_Ref Property

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

BTL - 7.3.2.X46.6 - Energy_Meter_Ref Property Test		
	Test Conditionality	This test must be executed if both Energy_Meter_Ref and Energy_Meter properties are present.
	Test Directives	
	Testing Hints	

3.60.5 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.X46.7 - CHANGE_OF_STATE for Passenger_Alarm (ConfirmedEventNotification)		
	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	
BTL - 7.3.2.X46.8 - CHANGE_OF_STATE for Passenger_Alarm (UnconfirmedEventNotification)		
	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	

3.60.6 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 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	
BTL - 7.3.1.X8.2 - Reliability_Evaluation_Inhibit Summarization Test		
	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	

3.60.7 Supports Reliability Evaluation

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

BTL - 8.4.X9.13 CHANGE_OF_RELIABILITY with FAULT_LISTED Algorithm (ConfirmedEventNotification)		
	Test Conditionality	This test must be executed
	Test Directives	
	Testing Hints	
BTL - 8.5.X9.14 CHANGE_OF_RELIABILITY with FAULT_LISTED Algorithm (UnconfirmedEventNotification)		
	Test Conditionality	This test must be executed
	Test Directives	
	Testing Hints	

3.60.8 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.X9.15 - CHANGE_OF_RELIABILITY FAULT-to-FAULT transitions in FAULT_LISTED		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

4.21 Data Sharing - WriteGroup - A

Devices claiming support for Data Sharing - WriteGroup - A must comply with the following section.

Checklist Changes

[In BTL Checklist, replace section Data Sharing - WriteGroup - A]

Data Sharing - WriteGroup - A		
	R~~[+]~~	Base Requirements
	~~[+]Contact BTL for interim tests for this BIBB.~~	

Test Plan Changes

[In BTL Test Plan, replace section 4.43 Data Sharing - WriteGroup - A]

4.21 Data Sharing - WriteGroup - A

4.21.1 Base Requirements

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

135.1-2013 - 8.X2.1 - Broadcasting to a Group of Channel Objects		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

4.27 Data Sharing - Life Safety View - A

Devices claiming support for Data Sharing - Life Safety View - A must comply with the following section.

Checklist Changes

[In BTL Checlist, modify section Data Sharing - Life Safety View - A]

Data Sharing - Life Safety View - A		
	R~~[+]~~	Base Requirements
	R	*Supports DS-RP-A*
	~~[+]Contact BTL for interim tests for this BIBB.~~	

Test Plan Changes

[In BTL Test Plan replace section 4.27 Data Sharing - Life Safety View - A]

4.27 Data Sharing - Life Safety View - A

4.27.1 Base Requirements

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

135.1-2013 - 8.18.3 - Reading and Presenting Properties		
	Test Conditionality	Must be executed if the IUT does not support DS-LSAV-A.
	Test Directives	Repeat the test for each of the standard object types and associated properties specified by DS-LSV-A.
	Testing Hints	

4.27.2 Supports DS-RP-A

The IUT shall support DS-RP-A in order to read properties for presentation.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-RP-A.
	Testing Hints	

4.28 Data Sharing - Life Safety Advanced View - A

Devices claiming support for Data Sharing - Life Safety Advanced View - A must comply with the following section.

Checklist Changes

[In BTL Checklist, replace section DS-LSAV-A]

Data Sharing - Life Safety Advanced View - A		
	R~~+~~	Base Requirements
	R	*Supports DS-RP-A*
	~~+Contact BTL for interim tests for this BIBB.~~	

Test Plan Changes

[In BTL Test Plan, replace 4.28 Data Sharing - Life Safety Advanced View - A]

4.28 Data Sharing - Life Safety Advanced View - A

4.28.1 Base Requirements

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

135.1-2013 - 8.18.3 - Reading and Presenting Properties		
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test for <u>all</u> standard objects and properties identified in DS-LSAV-A. For properties that contain a CHOICE construct, the IUT shall be capable of reading and presenting each of the forms of the datatype as defined in the IUT's claimed protocol revision. Full accuracy presentation is not required throughout the IUT, but there should be at least one place provided by the IUT that allows the presentation of each property to be presented in such a way that the presentation requirements of DS-LSAV-A are met.
	Testing Hints	

4.28.2 Supports DS-RP-A

The IUT shall support DS-RP-A in order to read properties for presentation.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-RP-A.
	Testing Hints	

4.29 Data Sharing - Life Safety Modify - A

Devices claiming support for Data Sharing - Life Safety Modify - A must comply with the following section.

Checklist Changes

[In BTL Checklist, replace section Data Sharing - Life Safety Modify - A]

Data Sharing - Life Safety Modify - A		
	R~~+~~	Base Requirements
	R	*Supports DS-WP-A*
	~~+Contact BTL for interim tests for this BIBB.~~	

Test Plan Changes

[In BTL Test Plan, replace section 4.29 Data Sharing - Life Safety Modify - A]

4.29 Data Sharing - Life Safety Modify - A

4.29.1 Base Requirements

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

135.1-2013 - 8.22.4 - Accepting Input and Modifying Properties		
	Test Conditionality	Must be executed if the IUT does not support DS-LSAM-A.
	Test Directives	Repeat the test for each of the required object types listed in the BIBB definition. Repeat for each of the required properties listed in the BIBB definition, except for those properties which are commandable. Repeat the test for a variety of values that cover the range of values required by the “Minimum Writable Value Ranges” table in the DS-M-A BIBB definition.
	Testing Hints	
135.1-2013 - 8.22.5 - Accepting Input and Commanding/Relinquishing Properties		
	Test Conditionality	Must be executed if the IUT does not support DS-LSAM-A.
	Test Directives	This test should be executed at priority 8 only, i.e. $PR_1 = 8$.
	Testing Hints	

4.29.2 Supports DS-WP-A

The IUT shall support DS-WP-A in order to update properties modified by the user.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-WP-A.
	Testing Hints	

4.30 Data Sharing - Life Safety Advanced Modify - A

Devices claiming support for Data Sharing - Life Safety Advanced Modify - A must comply with the following section.

Checklist Changes

[In BTL Checklist, replace section Data Sharing - Life Safety Advanced Modify - A]

Data Sharing - Life Safety Advanced Modify - A		
	R~~+~~	Base Requirements
	R	*Supports DS-WP-A*
	~~+Contact BTL for interim tests for this BIBB.~~	

Test Plan Changes

[Replace Test Plan Entry 4.30 Data Sharing - Life Safety Advanced Modify - A]

4.30 Data Sharing - Life Safety Advanced Modify - A

4.30.1 Base Requirements

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

135.1-2013 - 8.22.4 - Accepting Input and Modifying Properties		
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test for <u>each</u> of the required object types listed in the BIBB definition. Repeat for <u>each</u> of the required properties listed in the BIBB definition, except for those properties which are commandable. Repeat the test for a variety of values that cover the range of values required by the "Minimum Writable Value Ranges" table in the DS-M-A BIBB definition.
	Testing Hints	
135.1-2013 - 8.22.5 - Accepting Input and Commanding/Relinquishing Properties		
	Test Conditionality	Must be executed.
	Test Directives	This test should be executed at priority 8 only, i.e. $PR_1 = 8$.
	Testing Hints	

4.30.2 Supports DS-WP-A

The IUT shall support DS-WP-A in order to update properties modified by the user.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-WP-A.
	Testing Hints	

4.31 Data Sharing - Access Control View - A

Devices claiming support for Data Sharing - Access Control View - A must comply with the following section.

Checklist Changes

[In BTL Checklist, replace section Data Sharing - Access Control View - A]

Data Sharing - Access Control View - A		
	R~~[+]~~	Base Requirements
	R	*Supports DS-RP-A*
	~~[+]Contact BTL for interim tests for this BIBB.~~	

Test Plan Changes

[In BTL Test Plan, replace section 4.31 Data Sharing - Access Control View - A]

4.31 Data Sharing - Access Control View - A

4.31.1 Base Requirements

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

135.1-2013 - 8.18.3 - Reading and Presenting Properties		
	Test Conditionality	Must be executed if the IUT does not support DS-ACAV-A.
	Test Directives	Repeat the test for <u>each</u> of the standard object types and associated properties specified by DS-ACV-A.
	Testing Hints	

4.31.2 Supports DS-RP-A

The IUT shall support DS-RP-A in order to read properties for presentation.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-RP-A.
	Testing Hints	

4.32 Data Sharing - Access Control Advanced View - A

Devices claiming support for Data Sharing - Access Control Advanced View - A must comply with the following section.

Checklist Changes

[In BTL Checklist, replace section Data Sharing - Access Control Advanced View - A]

Data Sharing - Access Control Advanced View - A		
	R~~†~~	Base Requirements
	R	*Supports DS-RP-A*
	~~†Contact BTL for interim tests for this BIBB.~~	

Test Plans Changes

[In BTL Test Plan, replace section 4.32 Data Sharing - Access Control Advanced View - A]

4.32 Data Sharing - Access Control Advanced View - A

4.32.1 Base Requirements

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

135.1-2013 - 8.18.3 - Reading and Presenting Properties		
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test for <u>all</u> standard objects and properties identified in DS-ACAV-A. For properties that contain a CHOICE construct, the IUT shall be capable of reading and presenting each of the forms of the datatype as defined in the IUT's claimed protocol revision. Full accuracy presentation is not required throughout the IUT, but there should be at least one place provided by the IUT that allows the presentation of each property to be presented in such a way that the presentation requirements of DS-ACAV-A are met.
	Testing Hints	

4.32.2 Supports DS-RP-A

The IUT shall support DS-RP-A in order to read properties for presentation.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-RP-A.
	Testing Hints	

4.33 Data Sharing - Access Control Modify - A

Devices claiming support for Data Sharing - Access Control Modify - A must comply with the following section.

Checklist Changes

[In BTL Checklist, replace section Data Sharing - Access Control Modify - A]

Data Sharing - Access Control Modify - A		
	~~R[‡]~~	Base Requirements
	R	*Supports DS-WP-A*
	~~[‡]Contact BTL for interim tests for this BIBB.~~	

Test Plans Changes

[In BTL Test Plan, replace section 4.32 Data Sharing - Access Control Advanced View - A]

4.33 Data Sharing - Access Control Modify - A

4.33.1 Base Requirements

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

135.1-2013 - 8.22.4 - Accepting Input and Modifying Properties		
	Test Conditionality	Must be executed if the IUT does not support DS-ACAM-A.
	Test Directives	Repeat the test for <u>each</u> of the required object types listed in the BIBB definition. Repeat for <u>each</u> of the required properties listed in the BIBB definition, except for those properties which are commandable. Repeat the test for a variety of values that cover the range of values required by the "Minimum Writable Value Ranges" table in the DS-M-A BIBB definition.
	Testing Hints	
135.1-2013 - 8.22.5 - Accepting Input and Commanding/Relinquishing Properties		
	Test Conditionality	Must be executed if the IUT does not support DS-ACAM-A.
	Test Directives	This test should be executed at priority 8 only, i.e. $PR_1 = 8$.
	Testing Hints	

4.33.2 Supports DS-WP-A

The IUT shall support DS-WP-A in order to update properties modified by the user.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-WP-A.
	Testing Hints	

4.34 Data Sharing - Life Safety Advanced Modify - A

Devices claiming support for Data Sharing - Access Control Advanced Modify - A must comply with the following section.

Checklist Changes

[In BTL Checklist, replace section Data Sharing - Access Control Advanced Modify - A]

Data Sharing - Access Control Advanced Modify - A		
	R~~+~~	Base Requirements
	R	*Supports DS-WP-A*
	~~+Contact BTL for interim tests for this BIBB.~~	

Test Plan Changes

[In BTL Test Plan, replace section 4.34 Data Sharing - Life Safety Advanced Modify - A]

4.34 Data Sharing - Life Safety Advanced Modify - A

4.34.1 Base Requirements

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

135.1-2013 - 8.22.4 - Accepting Input and Modifying Properties		
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test for <u>each</u> of the required object types listed in the BIBB definition. Repeat for <u>each</u> of the required properties listed in the BIBB definition, except for those properties which are commandable. Repeat the test for a variety of values that cover the range of values required by the “Minimum Writable Value Ranges” table in the DS-M-A BIBB definition.
	Testing Hints	
135.1-2013 - 8.22.5 - Accepting Input and Commanding/Relinquishing Properties		
	Test Conditionality	Must be executed.
	Test Directives	This test should be executed at priority 8 only, i.e. $PR_1 = 8$.
	Testing Hints	

4.34.2 Supports DS-WP-A

The IUT shall support DS-WP-A in order to update properties modified by the user.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-WP-A.
	Testing Hints	

4.35 Data Sharing - Access Control User Configuration - A

Devices claiming support for Data Sharing - Access Control User Configuration - A must comply with the following section.

Checklist Changes

[In BTL Checklist, replace section Data Sharing - Access Control User Configuration - A]

Data Sharing - Access Control User Configuration - A		
	R~~[1]~~	Base Requirements
	R	*Supports DS-RP-A*
	R	*Supports DS-WP-A*
	R	*Supports DM-OCD-A*
	~~[1]Contact BTL for interim tests for this BIBB.~~	

Test Plan Changes

[In BTL Test Plan, replace section 4.35 Data Sharing - Access Control User Configuration - A]

4.35 Data Sharing - Access Control User Configuration - A

4.35.1 Base Requirements

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

135.1-2013 - 8.18.3 - Reading and Presenting Properties		
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test for <u>each</u> of the standard object types and associated properties specified by DS-ACUC-A.
	Testing Hints	
135.1-2013 - 8.22.4 - Accepting Input and Modifying Properties		
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test for <u>each</u> of the required object types listed in the BIBB definition. Repeat for <u>each</u> of the required properties listed in the BIBB definition, except for those properties which are commandable. Repeat the test for a variety of values that cover the range of values required by the "Minimum Writable Value Ranges" table in the DS-M-A BIBB definition.
	Testing Hints	
135.1-2013 - 8.22.5 - Accepting Input and Commanding/Relinquishing Properties		
	Test Conditionality	Must be executed.
	Test Directives	This test should be executed at priority 8 only, i.e. $PR_1 = 8$.
	Testing Hints	

4.35.2 Supports DS-RP-A

The IUT shall support DS-RP-A in order to read properties of Access Control objects.

Verify Checklist

	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-RP-A.
	Testing Hints	

4.35.3 Supports DS-WP-A

The IUT shall support DS-WP-A in order to update properties modified by the user.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-WP-A.
	Testing Hints	

4.35.2 Supports DM-OCD-A

The IUT shall support DM-OCD-A in order to create and delete Access Control objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DM-OCD-A, and that all object types required by DS-ACUC-A are claimed within DM-OCD-A.
	Testing Hints	

4.37 Data Sharing - Access Control Site Configuration - A

Devices claiming support for Data Sharing - Access Control Site Configuration - A must comply with the following section.

Checklist Changes

[In BTL Checklist, replace section Data Sharing - Access Control Site Configuration - A]

Data Sharing - Access Control Site Configuration - A		
	R~~+~~	Base Requirements
	R	*Supports DS-RP-A*
	R	*Supports DS-WP-A*
	R	*Supports DM-OCD-A*
	~~+Contact BTL for interim tests for this BIBB.~~	

Test Plan Changes

[In BTL Test Plan, replace section 4.37 Data Sharing - Access Control Site Configuration - A]

4.37 Data Sharing - Access Control Site Configuration - A

4.37.1 Base Requirements

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

135.1-2013 - 8.18.3 - Reading and Presenting Properties		
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test for <u>each</u> of the standard object types and associated properties specified by DS-ACSC-A.
	Testing Hints	
135.1-2013 - 8.22.4 - Accepting Input and Modifying Properties		
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test for <u>each</u> of the required object types listed in the BIBB definition. Repeat for <u>each</u> of the required properties listed in the BIBB definition, except for those properties which are commandable. Repeat the test for a variety of values that cover the range of values required by the “Minimum Writable Value Ranges” table in the DS-M-A BIBB definition.
	Testing Hints	
135.1-2013 - 8.22.5 - Accepting Input and Commanding/Relinquishing Properties		
	Test Conditionality	Must be executed.
	Test Directives	This test should be executed at priority 8 only, i.e. $PR_1 = 8$.
	Testing Hints	

4.37.2 Supports DS-RP-A

The IUT shall support DS-RP-A in order to read properties of Access Control objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-RP-A.
	Testing Hints	

4.37.3 Supports DS-WP-A

The IUT shall support DS-WP-A in order to update properties modified by the user.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-WP-A.
	Testing Hints	

4.37.2 Supports DM-OCD-A

The IUT shall support DM-OCD-A in order to create and delete Access Control objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DM-OCD-A, and that all object types required by DS-ACSC-A are claimed within DM-OCD-A.
	Testing Hints	

4.40 Data Sharing - Access Control Access Door - A

Devices claiming support for Data Sharing - Access Control Access Door - A must comply with the following section.

Checklist Changes

[In BTL Checklist, replace section Data Sharing - Access Control Access Door - A]

Data Sharing - Access Control Access Door - A		
	R[+]	Base Requirements
	R	*Supports DS-RP-A*
	R	*Supports DS-WP-A*
	~~[+]Contact BTL for interim tests for this BIBB.~~	

Test Plan Changes

[In BTL Test Plan, replace section 4.40 Data Sharing - Access Control Access Door - A]

4.40 Data Sharing - Access Control Access Door - A

4.40.1 Base Requirements

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

135.1-2013 - 8.18.3 - Reading and Presenting Properties		
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test for <u>each</u> of the standard object types and associated properties specified by DS-ACAD-A.
	Testing Hints	
135.1-2013 - 8.22.4 - Accepting Input and Modifying Properties		
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test for <u>each</u> of the required object types listed in the BIBB definition. Repeat for <u>each</u> of the required properties listed in the BIBB definition, except for those properties which are commandable. Repeat the test for a variety of values that cover the range of values required by the "Minimum Writable Value Ranges" table in the DS-M-A BIBB definition.
	Testing Hints	
135.1-2013 - 8.22.5 - Accepting Input and Commanding/Relinquishing Properties		
	Test Conditionality	Must be executed.
	Test Directives	This test should be executed at priority 8 only, i.e. $PR_1 = 8$.
	Testing Hints	

4.40.2 Supports DS-RP-A

The IUT shall support DS-RP-A in order to read properties of Access Door objects.

Verify Checklist		
	Test Conditionality	Must be executed.

	Test Directives	Verify that the IUT claims support for DS-RP-A.
	Testing Hints	

4.40.3 Supports DS-WP-A

The IUT shall support DS-WP-A in order to update Access Door properties modified by the user.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-WP-A.
	Testing Hints	

4.41 Data Sharing - Access Control Credential Data Input - A

Devices claiming support for Data Sharing - Access Control Credential Data Input - A must comply with the following section.

Checklist Changes

[In BTL Checklist, replace section Data Sharing - Access Control Credential Data Input - A]

Data Sharing - Access Control Credential Data Input - A		
	R[+]	Base Requirements
	R	*Supports DS-RP-A*
	R	*Supports DS-WP-A*
	R	*Supports DS-COV-A*
	[+]~~Contact BTL for interim tests for this BIBB.~~	

Test Plan Changes

[In BTL Test Plan, replace section 4.41 Data Sharing - Access Control Credential Data Input - A]

4.41 Data Sharing - Access Control Credential Data Input - A

4.41.1 Base Requirements

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

135.1-2013 - 8.18.3 - Reading and Presenting Properties		
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test for <u>each</u> of the standard object types and associated properties specified by DS-ACCDI-A.
	Testing Hints	
135.1-2013 - 8.22.4 - Accepting Input and Modifying Properties		
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test for <u>each</u> of the required object types listed in the BIBB definition. Repeat for <u>each</u> of the required properties listed in the BIBB definition, except for those properties which are commandable. Repeat the test for a variety of values that cover the range of values required by the "Minimum Writable Value Ranges" table in the DS-M-A BIBB definition.
	Testing Hints	
135.1-2013 - 8.22.5 - Accepting Input and Commanding/Relinquishing Properties		
	Test Conditionality	Must be executed.
	Test Directives	This test should be executed at priority 8 only, i.e. $PR_1 = 8$.
	Testing Hints	

4.41.2 Supports DS-RP-A

The IUT shall support DS-RP-A in order to read properties of Credential Data Input objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-RP-A.
	Testing Hints	

4.41.3 Supports DS-WP-A

The IUT shall support DS-WP-A in order to update Credential Data Input properties modified by the user.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-WP-A.
	Testing Hints	

4.41.4 Supports DS-COV-A

The IUT shall support DS-COV-A in order to receives COV notifications for Credential Data Input objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-COV-A, and that Credential Data Input is claimed within DM-COV-A.
	Testing Hints	

4.43 Data Sharing - Lighting Output - A

Devices claiming support for Data Sharing - Lighting Output - A must comply with the following section.

Checklist Changes

[In BTL Checklist, replace section Data Sharing - Lighting Output - A]

Data Sharing - Lighting Output - A		
	R~~†~~	Base Requirements
	R	*Supports DS-WP-A*
~~†Contact BTL for interim tests for this BIBB.~~		

Test Plan Changes

[In BTL Test Plan, replace section 4.43 Data Sharing - Lighting Output - A]

4.43 Data Sharing - Lighting Output - A

4.43.1 Base Requirements

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

135.1-2013 - 8.22.1 - Writing Non-Array Properties		
	Test Conditionality	Must be executed if the IUT does not support DS-ALO-A.
	Test Directives	Repeat the test for each of the object types listed in the BIBB, writing to the Present_Value property.
	Testing Hints	

4.43.2 Supports DS-WP-A

The IUT shall support DS-WP-A in order to control objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-WP-A.
	Testing Hints	

4.44 Data Sharing - Lighting Output Status - A

Devices claiming support for Data Sharing - Lighting Output Status - A must comply with the following section.

Checklist Changes

[In BTL Checklist, replace section Data Sharing - Lighting Output Status - A]

Data Sharing - Lighting Output Status - A		
	R~~+~~	Base Requirements
	R	*Supports DS-RP-A*
	~~+Contact BTL for interim tests for this BIBB.~~	

Test Plan Changes

[In BTL Test Plan, replace section 4.44 Data Sharing - Lighting Output Status - A]

4.44 Data Sharing - Lighting Output Status - A

4.44.1 Base Requirements

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

135.1-2013 - 8.18.1 - Reading Non-Array Properties		
	Test Conditionality	
	Test Directives	Repeat the test for each of the object types listed in the BIBB, reading the Present_Value and Egress_Active properties from the objects types as required by the BIBB.
	Testing Hints	

4.44.2 Supports DS-RP-A

The IUT shall support DS-RP-A in order to retrieve property values from lighting objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-RP-A, and claims the ability to read non-array properties, Enumerated, Unsigned, and REAL properties.
	Testing Hints	

4.45 Data Sharing - Advanced Lighting Output - A

Devices claiming support for Data Sharing - Advanced Lighting Output - A must comply with the following section.

Checklist Changes

[In BTL Checklist, replace section Data Sharing - Advanced Lighting Output - A]

Data Sharing - Advanced Lighting Output - A		
	R~~+~~	Base Requirements
	R	*Supports DS-WP-A*
	~~+Contact BTL for interim tests for this BIBB.~~	

Test Plan Changes

[In BTL Test Plan, replace section 4.45 Data Sharing - Advanced Lighting Output - A]

4.45 Data Sharing - Advanced Lighting Output - A

4.45.1 Base Requirements

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

135.1-2013 - 8.22.1 - Writing Non-Array Properties		
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test for each property of each of the object types listed in the BIBB, except those that are required to be read-only by the standard.
	Testing Hints	

4.45.2 Supports DS-WP-A

The IUT shall support DS-WP-A in order to control objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-WP-A.
	Testing Hints	

4.48 Data Sharing - Lighting Output Management - A

Devices claiming support for Data Sharing - Lighting Output Management - A must comply with the following section.

Checklist Changes

[In BTL Checklist, replace section Data Sharing - Lighting Output Management - A]

Data Sharing - Lighting Output Management - A		
	R~~†~~	Base Requirements
	R	*Supports DM-OCD-A*
~~†Contact BTL for interim tests for this BIBB.~~		

Test Plan Changes

[In BTL Test Plan, replace section 4.48 Data Sharing - Lighting Output Management - A]

4.48 Data Sharing - Lighting Output Management - A

4.48.1 Base Requirements

Base requirements must be met by any IUT claiming conformance to this BIBB. There are no base requirements tests for this section.

4.48.2 Supports DM-OCD-A

The IUT shall support DM-OCD-A in order to create and delete Access Control objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DM-OCD-A, and that all object types required by DS-LOM-A are claimed within DM-OCD-A.
	Testing Hints	

4.49 Data Sharing - Lighting View - A

Devices claiming support for Data Sharing - Lighting View - A must comply with the following section.

Checklist Changes

[In BTL Checklist, replace section Data Sharing - Lighting View - A]

Data Sharing - Lighting View - A		
	R~~+~~	Base Requirements
	R	*Supports DS-RP-A*
	~~+Contact BTL for interim tests for this BIBB.~~	

Test Plan Changes

[In BTL Test Plan, replace section 4.49 Data Sharing - Lighting View - A]

4.49 Data Sharing - Lighting View - A

4.49.1 Base Requirements

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

135.1-2013 - 8.18.3 - Reading and Presenting Properties		
	Test Conditionality	Must be executed if the IUT does not support DS-LAV-A.
	Test Directives	Repeat the test for each of the standard object types and associated properties specified by DS-LV-A.
	Testing Hints	

4.49.2 Supports DS-RP-A

The IUT shall support DS-RP-A in order to read properties for presentation.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-RP-A.
	Testing Hints	

4.50 Data Sharing - Lighting Advanced View - A

Devices claiming support for Data Sharing - Lighting Advanced View - A must comply with the following section.

Checklist Changes

[In BTL Checklist, replace section Data Sharing - Lighting Advanced View - A]

Data Sharing - Lighting Advanced View - A		
	R~~+~~	Base Requirements
	R	*Supports DS-RP-A*
	~~+Contact BTL for interim tests for this BIBB.~~	

Test Plan Changes

[In BTL Test Plan, replace section 4.50 Data Sharing - Lighting Advanced View - A]

4.50 Data Sharing - Lighting Advanced View - A

4.50.1 Base Requirements

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

135.1-2013 - 8.18.3 - Reading and Presenting Properties		
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test for <u>all</u> standard objects and properties identified in DS-LAV-A. For properties that contain a CHOICE construct, the IUT shall be capable of reading and presenting each of the forms of the datatype as defined in the IUT's claimed protocol revision. Full accuracy presentation is not required throughout the IUT, but there should be at least one place provided by the IUT that allows the presentation of each property to be presented in such a way that the presentation requirements of DS-LAV-A are met.
	Testing Hints	

4.50.2 Supports DS-RP-A

The IUT shall support DS-RP-A in order to read properties for presentation.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-RP-A.
	Testing Hints	

4.51 Data Sharing - Lighting Modify - A

Devices claiming support for Data Sharing - Lighting Modify - A must comply with the following section.

Checklist Changes

[In BTL Checklist, replace section Data Sharing - Lighting Modify - A]

Data Sharing - Lighting Modify - A		
	R~~+~~	Base Requirements
	R	*Supports DS-WP-A*
	~~+Contact BTL for interim tests for this BIBB.~~	

Test Plan Changes

[In BTL Test Plan, replace section 4.51 Data Sharing - Lighting Modify - A]

4.51 Data Sharing - Lighting Modify - A

4.51.1 Base Requirements

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

135.1-2013 - 8.22.4 - Accepting Input and Modifying Properties		
	Test Conditionality	Must be executed if the IUT does not support DS-LAM-A.
	Test Directives	Repeat the test for <u>each</u> of the required object types listed in the BIBB definition. Repeat for <u>each</u> of the required properties listed in the BIBB definition, except for those properties which are commandable. Repeat the test for a variety of values that cover the range of values required by the “Minimum Writable Value Ranges” table in the DS-M-A BIBB definition.
	Testing Hints	
135.1-2013 - 8.22.5 - Accepting Input and Commanding/Relinquishing Properties		
	Test Conditionality	Must be executed if the IUT does not support DS-LAM-A.
	Test Directives	This test should be executed at priority 8 only, i.e. $PR_1 = 8$.
	Testing Hints	

4.51.2 Supports DS-WP-A

The IUT shall support DS-WP-A in order to update properties modified by the user.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-WP-A.
	Testing Hints	

4.52 Data Sharing - Lighting Advanced Modify - A

Devices claiming support for Data Sharing - Lighting Advanced Modify - A must comply with the following section.

Checklist Changes

[In BTL Checklist, replace section Data Sharing - Lighting Advanced Modify - A]

Data Sharing - Lighting Advanced Modify - A		
	~~R+~~ R	Base Requirements
	R	*Supports DS-WP-A*
	~~+Contact BTL for interim tests for this BIBB.~~	

Test Plan Changes

[In BTL Test Plan, replace section 4.52 Data Sharing - Lighting Advanced Modify - A]

4.52 Data Sharing - Lighting Advanced Modify - A

4.52.1 Base Requirements

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

135.1-2013 - 8.22.4 - Accepting Input and Modifying Properties		
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test for <u>each</u> of the required object types listed in the BIBB definition. Repeat for <u>each</u> of the required properties listed in the BIBB definition, except for those properties which are commandable. Repeat the test for a variety of values that cover the range of values required by the "Minimum Writable Value Ranges" table in the DS-M-A BIBB definition.
	Testing Hints	
135.1-2013 - 8.22.5 - Accepting Input and Commanding/Relinquishing Properties		
	Test Conditionality	Must be executed.
	Test Directives	This test should be executed at priority 8 only, i.e. $PR_1 = 8$.
	Testing Hints	

4.52.2 Supports DS-WP-A

The IUT shall support DS-WP-A in order to update properties modified by the user.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-WP-A.
	Testing Hints	

5.27 Alarm and Event Management - Life Safety View Notifications - A

Devices claiming support for Alarm and Event Management - Life Safety View Notification - A must comply with the following section.

Checklist Changes

[In BTL Checklist, replace section Alarm and Event Management - Life Safety View Notifications - A]

Alarm and Event Management - Life Safety View Notifications - A		
	R~~¹~~	Base Requirements
	R	*Supports AE-N-A*
	R	*Supports AE-LS-A*
	~~¹Contact BTL for interim tests for this BIBB.~~	

Test Plan Changes

[In BTL Test Plan, replace section 5.27 Alarm and Event Management - Life Safety View Notifications - A]

5.27 Alarm and Event Management - Life Safety View Notifications - A

5.27.1 Base Requirements

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

BTL - 9.4.5 - ConfirmedEventNotification Simple Presentation		
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test for CHANGE_OF_LIFE_SAFETY, and each of the transitions defined for that event type. Repeat the test for FAULT_LIFE_SAFETY. Execute at least once with a Message_Text 32 or more characters in length.
	Testing Hints	
135.1-2013 - 9.5.1 - UnconfirmedEventNotification Simple Presentation		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	Repeat the test for CHANGE_OF_LIFE_SAFETY, and each of the transitions defined for that event type. Repeat the test for FAULT_LIFE_SAFETY. Execute at least once with a Message_Text 32 or more characters in length.

5.27.2 Supports AE-N-A

The IUT shall support AE-N-A in order to receive and display event notifications.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-N-A.

	Testing Hints	

5.27.3 Supports AE-LS-A

The IUT shall support AE-LS-A in order to silence / unsilence life safety objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-LS-A.
	Testing Hints	

5.28 Alarm and Event Management - Life Safety Advanced View Notifications - A

Devices claiming support for Alarm and Event Management - Life Safety Advanced View Notifications - A must comply with the following section.

Checklist Changes

[In BTL Checklist, replace section Alarm and Event Management - Life Safety Advanced View Notifications - A]

Alarm and Event Management - Life Safety Advanced View Notifications - A		
	R~~+~~	Base Requirements
	R	Supports AE-AVN-A
	R	Supports AE-LS-A
	~~+Contact BTL for interim tests for this BIBB.~~	

Test Plan Changes

[In BTL Test Plan, replace section 5.28 Alarm and Event Management - Life Safety Advanced View Notifications - A]

5.28 Alarm and Event Management - Life Safety Advanced View Notifications - A

5.28.1 Base Requirements

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

BTL - 9.4.6 - ConfirmedEventNotification Full Presentation		
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test for CHANGE_OF_LIFE_SAFETY, and each of the transitions defined for that event type. Repeat the test for FAULT_LIFE_SAFETY. Execute at least once with a Message_Text 256 or more characters in length.
	Testing Hints	
135.1-2013 - 9.5.2 - UnconfirmedEventNotification Full Presentation		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	Repeat the test for CHANGE_OF_LIFE_SAFETY, and each of the transitions defined for that event type. Repeat the test for FAULT_LIFE_SAFETY. Execute at least once with a Message_Text 256 or more characters in length.

5.28.2 Supports AE-AVN-A

The IUT shall support AE-AVN-A in order to receive and display standard event notifications for most standard object types.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-AVN-A.
	Testing Hints	

5.28.3 Supports AE-LS-A

The IUT shall support AE-LS-A in order to silence / unsilence life safety objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-LS-A.
	Testing Hints	

5.29 Alarm and Event Management - Life Safety View Modify - A

Devices claiming support for Alarm and Event Management - Life Safety View Modify - A must comply with the following section.

Checklist Changes

[In BTL Checklist, replace section Alarm and Event Management - Life Safety View Modify - A]

Alarm and Event Management - Life Safety View Modify - A		
	R~~+~~	Base Requirements
	R	*Supports DS-RP-A*
	R	*Supports DS-WP-A*
	R	*Supports AE-VM-A*
	~~+Contact BTL for interim tests for this BIBB.~~	

Test Plan Changes

[In BTL Test Plan, replace section 5.29 Alarm and Event Management - Life Safety View Modify - A]

5.29 Alarm and Event Management - Life Safety View Modify - A

5.29.1 Base Requirements

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

135.1-2013 - 8.18.3 - Reading and Presenting Properties		
	Test Conditionality	Must be executed if AE-LSAVM-A is not supported.
	Test Directives	Repeat the test for each standard object capable of generating CHANGE_OF_LIFE_SAFETY events, reading and displaying the pAlarmValues and pLifeSafetyAlarmValues properties. Repeat the test for each standard object capable of using the FAULT_LIFE_SAFETY algorithm, reading and displaying the pFaultValues property.
	Testing Hints	
135.1-2013 - 8.22.4 - Accepting Input and Modifying Properties		
	Test Conditionality	Must be executed if AE-LSAVM-A is not supported.
	Test Directives	Repeat the test for each standard object capable of generating CHANGE_OF_LIFE_SAFETY events, reading and displaying the pAlarmValues and pLifeSafetyAlarmValues properties. Repeat the test for each standard object capable of using the FAULT_LIFE_SAFETY algorithm, reading and displaying the pFaultValues property.
	Testing Hints	

5.29.2 Supports DS-RP-A

The IUT shall support DS-RP-A in order to read properties for presentation.

Verify Checklist		
	Test Conditionality	Must be executed.

	Test Directives	Verify that the IUT claims support for DS-RP-A.
	Testing Hints	

5.29.3 Supports DS-WP-A

The IUT shall support DS-WP-A in order to update properties modified by the user.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-WP-A.
	Testing Hints	

5.29.4 Supports AE-VM-A

The IUT shall support AE-VM-A in order to facilitate configuration of alarm parameters by the user.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-VM-A.
	Testing Hints	

5.30 Alarm and Event Management - Life Safety Advanced View Modify - A

Devices claiming support for Alarm and Event Management - Life Safety Advanced Modify - A must comply with the following section.

Checklist Changes

[In BTL Checklist, replace section Alarm and Event Management - Life Safety Advanced View Modify - A]

Alarm and Event Management - Life Safety Advanced View Modify - A		
	R[+]	Base Requirements
	R	*Supports DS-RP-A*
	R	*Supports DS-WP-A*
	R	*Supports DM-OCD-A*
	R	*Supports AE-AVM-A*
	[+]~~Contact BTL for interim tests for this BIBB.~~	

Test Plan Changes

[In BTL Test Plan, replace section 5.30 Alarm and Event Management - Life Safety Advanced View Modify - A]

5.30 Alarm and Event Management - Life Safety Advanced View Modify - A

5.30.1 Base Requirements

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

135.1-2013 - 8.18.3 - Reading and Presenting Properties		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	Repeat the test for each standard event generating object type which can generate CHANGE_OF_LIFE_SAFETY event notifications, or use the FAULT_LIFE_SAFETY algorithm.
135.1-2013 - 8.22.4 - Accepting Input and Modifying Properties		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	Repeat the test for each standard event generating object type which can generate CHANGE_OF_LIFE_SAFETY event notifications, or use the FAULT_LIFE_SAFETY algorithm.

5.30.2 Supports DS-RP-A

The IUT shall support DS-RP-A in order to read properties for presentation.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-RP-A.
	Testing Hints	

5.30.3 Supports DS-WP-A

The IUT shall support DS-WP-A in order to update properties modified by the user.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-WP-A.
	Testing Hints	

5.30.4 Supports DM-OCD-A

The IUT shall support DM-OCD-A in order to facilitate creation and deletion of life safety objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DM-OCD-A and that all object types required by DS-LSAVM-A are claimed within DM-OCD-A.
	Testing Hints	

5.30.5 Supports AE-AVM-A

The IUT shall support AE-AVM-A in order to facilitate configuration of alarm parameters by the user.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-AVM-A.
	Testing Hints	

5.31 Alarm and Event Management - Access Control - A

Devices claiming support for Alarm and Event Management - Access Control - A must comply with the following section.

Checklist Changes

[In BTL Checklist, replace section Alarm and Event Management - Access Control - A]

Alarm and Event Management - Access Control - A		
	R	Base Requirements
	R	Executes ConfirmedEventNotifications
	R	Executes UnconfirmedEventNotifications
	R	Processes intrinsically generated notifications
	R	Processes algorithmically generated notifications
	R	Processes event notifications with timestamps of the BACnetDateTime form
	R	Processes event notifications with timestamps of the Time form
	R	Processes event notifications with timestamps of the Sequence Number form
	R	Supports AE-ACK-A

Test Plan Changes

[In BTL Test Plan, replace section 5.31 Alarm and Event Management - Access Control - A]

5.31 Alarm and Event Management - Access Control - A

5.31.1 Base Requirements

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

BTL - 9.4.X1 - Unsupported Message Text Character Set ConfirmedEventNotification Test		
	Test Conditionality	If the IUT supports all character sets, this test shall be skipped.
	Test Directives	
	Testing Hints	
BTL - 9.5.X1 - Unsupported Message Text Character Set UnconfirmedEventNotification Test		
	Test Conditionality	If the IUT supports all character sets, this test shall be skipped.
	Test Directives	
	Testing Hints	

5.31.2 Executes ConfirmedEventNotifications

The IUT is capable of executing ConfirmedEventNotifications with an Event Type of ACCESS_EVENT. This functionality will be covered by the testing of the individual algorithms.

No Specific Test		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT's EPICS claims that it supports the ConfirmedEventNotification service.
	Testing Hints	

5.31.3 Executes UnconfirmedEventNotifications

The IUT is capable of executing UnconfirmedEventNotifications with an Event Type of ACCESS_EVENT. There are currently no tests defined for this functional item.

No Specific Test		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT's EPICS claims that it supports the UnconfirmedEventNotification service.
	Testing Hints	

5.31.4 Processes Intrinsically Generated Notifications

The IUT is capable of executing ConfirmedEventNotifications with an Event Type of ACCESS_EVENT that reference an object type other than Event Enrollment.

135.1-2013 - 9.4.1 - ConfirmedEventNotification Using the Time Form of the 'Timestamp' Parameter and Conveying a Text Message, 135.1-2013 - 9.4.2 - ConfirmedEventNotification Using the DateTime Form of the 'Timestamp' Parameter and no Text Message, or 135.1-2013 - 9.4.3 - ConfirmedEventNotification Using the Sequence Number Form of the 'Timestamp' Parameter and no Text Message		
	Test Conditionality	At least one of the tests must be executed with the Event Object Identifier referencing a BACnet object other than an Event Enrollment object.
	Test Directives	Execute using an event type of ACCESS_EVENT.
	Testing Hints	

5.31.5 Processes Algorithmically Generated Notifications

The IUT is capable of executing ConfirmedEventNotifications with an Event Type of ACCESS_EVENT that reference an Event Enrollment object.

135.1-2013 - 9.4.1 - ConfirmedEventNotification Using the Time Form of the 'Timestamp' Parameter and Conveying a Text Message, 135.1-2013 - 9.4.2 - ConfirmedEventNotification Using the DateTime Form of the 'Timestamp' Parameter and no Text Message, or 135.1-2013 - 9.4.3 - ConfirmedEventNotification Using the Sequence Number Form of the 'Timestamp' Parameter and no Text Message		
	Test Conditionality	At least one of the tests must be executed with the Event Object Identifier referencing an Event Enrollment object.
	Test Directives	Execute using an event type of ACCESS_EVENT.
	Testing Hints	

5.31.6 Processes Event Notifications with Timestamps of the BACnetDateTime Form

The IUT is capable of executing ConfirmedEventNotifications that contain a timestamp of the BACnetDateTime form.

135.1-2013 - 9.4.2 - ConfirmedEventNotification Using the DateTime Form of the 'Timestamp' Parameter and no Text Message		
	Test Conditionality	Must be executed.
	Test Directives	Execute using an event type of ACCESS_EVENT.
	Testing Hints	

5.31.7 Processes Event Notifications with Timestamps of the Time Form

The IUT is capable of executing ConfirmedEventNotifications that contain a timestamp of the Time form.

135.1-2013 - 9.4.1 - ConfirmedEventNotification Using the Time Form of the 'Timestamp' Parameter and Conveying a Text Message		
	Test Conditionality	Must be executed.
	Test Directives	Execute using an event type of ACCESS_EVENT.
	Testing Hints	

5.31.8 Processes Event Notifications with Timestamps of the Sequence Number Form

The IUT is capable of executing ConfirmedEventNotifications that contain a timestamp of the Sequence Number form.

135.1-2013 - 9.4.3 - ConfirmedEventNotification Using the Sequence Number Form of the 'Timestamp' Parameter and no Text Message		
	Test Conditionality	Must be executed.
	Test Directives	Execute using an event type of ACCESS_EVENT.
	Testing Hints	

5.31.9 Supports AE-ACK-A

The IUT must support AE-ACK-A if it claims support for AE-AC-A.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-ACK-A in the Checklist.
	Testing Hints	
BTL - 8.1 - ACKNOWLEDGEALARM SERVICE INITIATION TESTS		
	TEST CONDITIONALITY	Must be executed.
	Test Directives	Execute using an event type of ACCESS_EVENT. Execute once to acknowledge a ConfirmedEventNotification, and again to acknowledge an UnconfirmedEventNotification.
	TESTING HINTS	

5.33 Alarm and Event Management - Access Controls Advanced View Notifications - A

Devices claiming support for Alarm and Event Management - Access Control Advanced View Notifications - A must comply with the following section.

Checklist Changes

[In BTL Checklist, replace section Alarm and Event Management - Access Control Advanced View Notifications - A]

Alarm and Event Management - Access Control Advanced View Notifications - A		
	R~~+~~	Base Requirements
	R	*Supports AE-AVN-A*
	R	*Supports AE-AC-A*
~~+Contact BTL for interim tests for this BIBB.~~		

Test Plan Changes

[In BTL Test Plan, replace section 5.33 Alarm and Event Management - Access Controls Advanced View Notifications - A]

5.33 Alarm and Event Management - Access Controls Advanced View Notifications - A

5.33.1 Base Requirements

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

BTL - 9.4.6 - ConfirmedEventNotification Full Presentation		
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test for ACCESS_EVENT, and each of the transitions defined for that event type. Execute at least once with a Message_Text 256 or more characters in length.
	Testing Hints	
135.1-2013 - 9.5.2 - UnconfirmedEventNotification Full Presentation		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	Repeat the test for ACCESS_EVENT, and each of the transitions defined for that event type. Execute at least once with a Message_Text 256 or more characters in length.

5.33.2 Supports AE-AVN-A

The IUT must support AE-AVN-A in order to receive and display standard event notifications for most standard object types.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-AVN-A in the Checklist.

	Testing Hints	

5.33.3 Supports AE-AC-A

The IUT must support AE-AC-A if it claims support for AE-ACAVN-A.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-AC-A in the Checklist.
	Testing Hints	

5.34 Alarm and Event Management - Access Control View Modify - A

Devices claiming support for Alarm and Event Management - Access Control View Modify - A must comply with the following section.

Checklist Changes

[In BTL Checklist, replace section Alarm and Event Management - Access Control View Modify - A]

Alarm and Event Management - Access Control View Modify - A		
	R~~+~~	Base Requirements
	R	*Supports AE-VM-A*
	~~+Contact BTL for interim tests for this BIBB.~~	

Test Plan Changes

[In BTL Test Plan, replace section 5.34 Alarm and Event Management - Access Control View Modify - A]

5.34 Alarm and Event Management - Access Control View Modify - A

5.34.1 Base Requirements

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

135.1-2013 - 8.18.3 - Reading and Presenting Properties		
	Test Conditionality	Must be executed if AE-ACAVM-A is not supported.
	Test Directives	
	Testing Hints	Repeat the test for each standard object capable of generating ACCESS_EVENT events, reading and displaying the pAccessEvents and pAccessEventTime properties.
135.1-2013 - 8.22.4 - Accepting Input and Modifying Properties		
	Test Conditionality	Must be executed if AE-ACAVM-A is not supported.
	Test Directives	
	Testing Hints	Repeat the test for each standard object capable of generating ACCESS_EVENT events, reading and displaying the pAccessEvents and pAccessEventTime properties.

5.34.2 Supports AE-VM-A

The IUT shall support AE-VM-A in order to facilitate configuration of alarm parameters by the user.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-VM-A.
	Testing Hints	

5.35 Alarm and Event Management - Access Control Advanced View Modify - A

Devices claiming support for Alarm and Event Management - Access Control Advanced View Modify - A must comply with the following section.

Checklist Changes

[In BTL Checklist, replace section Alarm and Event Management - Access Control Advanced View Modify - A]

Alarm and Event Management - Access Control Advanced View Modify - A		
	R~~†~~	Base Requirements
	R	*Supports DS-RP-A*
	R	*Supports DS-WP-A*
	R	*Supports D-OCD-A*
	R	*Supports AE-AVM-A*
~~†Contact BTL for interim tests for this BIBB.~~		

Test Plan Changes

[In BTL Test Plan, replace section 5.35 Alarm and Event Management - Access Control Advanced View Modify - A]

5.35 Alarm and Event Management - Access Control Advanced View Modify - A

5.35.1 Base Requirements

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

135.1-2013 - 8.18.3 - Reading and Presenting Properties		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	Repeat the test for each standard event generating object type which can generate ACCESS_EVENT event notifications.
135.1-2013 - 8.22.4 - Accepting Input and Modifying Properties		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	Repeat the test for each standard event generating object type which can generate ACCESS_EVENT event notifications.

5.35.2 Supports DS-RP-A

The IUT shall support DS-RP-A in order to read properties for presentation.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-RP-A.
	Testing Hints	

5.35.3 Supports DS-WP-A

The IUT shall support DS-WP-A in order to update properties modified by the user.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-WP-A.
	Testing Hints	

5.35.4 Supports DM-OCD-A

The IUT shall support DM-OCD-A in order to facilitate creation and deletion of life safety objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DM-OCD-A and that all object types required by DS-ACAVM-A are claimed within DM-OCD-A.
	Testing Hints	

5.35.5 Supports AE-AVM-A

The IUT shall support AE-AVM-A in order to facilitate configuration of alarm parameters by the user.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AE-AVM-A.
	Testing Hints	

8.16 Device Management – ReinitializeDevice - B

See Test Package addendum 16.1ai for testing of ReinitializeDevice with Network Port objects.

8.30 Device Management – Slave Proxy - B

Devices claiming support for Device Management - Slave Proxy - B must claim support for Protocol_Revision 4 or higher and comply with the following section.

Addendum 135-2001*a* added MS/TP slave proxy functionality. This document makes needed changes in the BTL Test Package to claim the associated BIBB DM-SP-B.

These changes are not contained in any SSPC proposal.

Checklist Changes

[In BTL Checklist, replace Device Management - Slave Proxy - B section]

Device Management - Slave Proxy - B		
	R[1]	Base Requirements
	O	*Supports Automatic Slave Address Binding*
	[1]Contact BTL for interim tests for this BIBB.	

Test Plan Changes

[In BTL Test Plan, replace section 8.30 Device Management - Slave Proxy - B]

8.30 Device Management - Slave Proxy - B

8.30.1 Base Requirements

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

135.1-2013 - 13.5.1 Manual Slave Binding Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
135.1-2013 - 13.5.3 Proxy Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

8.30.2 Supports Automatic Slave Address Binding

The IUT support automatic slave address binding.

135.1-2013 - 13.5.2 Automatic Slave Discovery Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

9.* Data Link Layer - All

See Test Package addendum 16.1ai for changes related to Data Link Layer testing with Network Port objects.

9.4 BACnet/IP – Annex J - BBMD

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.

Test Plan Changes

[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 – 14.3.X1 - Write-BDT service is required to return Write-BDT-NAK		
	Test Conditionality	Must be executed in all devices claiming Protocol_Revision >= 17.
	Test Directives	
	Testing Hints	

9.9 Data Link Layer - IPv6

Addendum 135-2012aj added support for IPv6 in Protocol_Revision 18.

These changes are based on, and diverge from, SSPC proposal CLB-029.

Checklist Changes

[In BTL Checklist, replace Data Link Layer - IPv6 section]

Support	Listing	Option
Data Link Layer –IPv6		
	R	Base Requirements
	C[1]	Is able to operate in Normal mode
	C[1]	Is able to operate in Foreign mode
	C[2]	Is able to operate in BBMD mode
[1] Required if the device does not support BBMD mode. [2] Required if the device does not support Foreign mode.		

Test Plan Changes

[In BTL Test Plan, replace section 9.9 Data Link Layer - IPv6]

9.9 Data Link Layer - IPv6

9.9.1 Base Requirements

Base requirements must be met by any IUT that can act, or can be made to act, as a BACnet/IPv6 device in a non-BBMD mode.

BTL - 12.X.1.1 - Execute Original-Unicast-NPDU		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 12.X.1.2 - Execute Virtual-Address-Resolution		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

9.9.2 Is Able to Operate in Normal Mode

The IUT supports NORMAL mode.

BTL - 12.X.2.1.1 - Initiate Original-Broadcast-NPDU		
	Test Conditionality	If the IUT does not initiate broadcasts, this test shall be skipped.

	Test Directives	
	Testing Hints	
BTL - 12.X.2.1.2 - Execute Original-Broadcast-NPDU		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 12.X.2.1.3 - Execute Forwarded-NPDU		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 12.X.2.1.4 - Execute Address-Resolution		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 12.X.2.1.5 - Execute Forwarded-Address-Resolution		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 12.X.2.2.1 - Reject Register-Foreign-Device		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 12.X.2.2.2 - Reject Delete-Foreign-Device-Table-Entry		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 12.X.2.2.3 - Reject Distribute-Broadcast-To-Network		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

9.9.3 Is Able to Operate in Foreign Mode

The IUT supports FOREIGN mode.

BTL - 12.X.3.1.1 - Initiate Distribute-Broadcast-To-Network-NPDU		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 12.X.3.1.2 - Execute Forwarded-NPDU		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 12.X.3.1.3 - Execute Forwarded-Address-Resolution		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 12.X.3.2.1 - Ignores Original-Broadcast-NPDU		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 12.X.3.2.2 - Ignore Address-Resolution		
	Test Conditionality	Must be executed.

	Test Directives	
	Testing Hints	
BTL - 12.X.3.2.3 - Reject Register-Foreign-Device		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 12.X.3.2.4 - Reject Delete-Foreign-Device-Table-Entry		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 12.X.3.2.5- Reject Distribute-Broadcast-To-Network		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

9.9.4 Is Able to Operate in BBMD Mode

The IUT supports BBMD mode.

BTL - 12.X.4.1.1 - Original-Broadcast-NPDU		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 12.X.4.1.2 - Forwarded-NPDU		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 12.X.4.1.3 - Address-Resolution		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 12.X.4.1.4 - Forwarded-Address-Resolution		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 12.X.4.1.5 - Distribute-Broadcast-To-Network		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 12.X.4.2.1 - Reject Forwarded-NPDU		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 12.X.4.2.2 - Reject Address-Resolution		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 12.X.4.2.3 - Reject Forwarded-Address-Resolution		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 12.X.4.2.4 - Reject Distribute-Broadcast-To-Network		
	Test Conditionality	Must be executed.
	Test Directives	

	Testing Hints	
BTL - 12.X.4.3.1 - Verify writability of the BDT		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 12.X.5.1 - Execute Register-Foreign-Device		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 12.X.5.2 - Execute Delete-Foreign-Device-Table-Entry		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 12.X.5.3.1 - Non-Zero-Duration Foreign Device Table Timer Operations		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 12.X.5.3.2 - Zero-Duration Foreign Device Timer Operations		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 12.X.5.4 - Delete-Foreign-Device-Table-Entry For A Non-existent Entry		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

10.4 Network Management - Connection Establishment - B

There are no tests in the BTL Test Package for connection establishment BIBBs.

These changes are not contained in any SSPC proposal.

Checklist Changes

[In BTL Checklist, modify Network Management - Connection Establishment - B section]

Network Management - Connection Establishment - B		
	R~~1~~	Base Requirements
	~~1 Contact BTL for interim tests for this BIBB.~~	

Test Plan Changes

[In BTL Test Plan, replace section 10.4 Network Management - Connection Establishment - B]

10.4 Network Management - Connection Establishment - B

9.10.1 Base Requirements

Base requirements must be met by any IUT that supports NM-CE-B.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims Network Management - Routing
	Testing Hints	Note that when applying routing tests to a half-router, the PTP connection should be established before the tests are started, and the IUT plus its peer half-router are together considered the router under test.
135.1-2013 - 10.3.1.2 - A Network Number is Specified that can be Reached Through a PTP Connection		
	Test Conditionality	Must be executed.
	Test Directives	Configure the test network as per 10.2.
	Testing Hints	
135.1-2013 - 10.3.3 - Initiating Half-Router Procedure for Connection Establishment		
	Test Conditionality	Must be executed.
	Test Directives	Configure the test network as per 10.2.
	Testing Hints	
135.1-2013 - 10.3.7 - Disconnect-Connection-To-Network		
	Test Conditionality	Must be executed.
	Test Directives	Configure the test network as per 10.2.
	Testing Hints	

10.7 Network Management - BBMD Configuration - B

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.

Checklist Changes

[In BTL Checklist, replace Network Management - BBMD 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.		

Test Plan Changes

[In BTL Test Plan, replace section 10.7 Network Management - BBMD Configuration - B]

10.7 Network Management - BBMD 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.

10.7.1 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 Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	
BTL - 14.2.2.2 - Execute Original-Broadcast-NPDU (Two-hop Distribution)		
	Test Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	
135.1-2013 - 14.2.3 - Execute Original-Unicast-NPDU		

	Test Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	
135.1-2013 - 14.5.2.2 - Original-Broadcast-NPDU Which Shall Be Forwarded (Two-hop Distribution)		
	Test Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	
BTL - 14.7.1.2 - Broadcast Message from Directly Connected IP Subnet (Two-hop Distribution)		
	Test Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	
BTL - 14.7.2.2 - Broadcast Message Forwarded by a Peer BBMD (Two-hop Distribution)		
	Test Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	
135.1-2013 - 14.9.3 - Original-Broadcast-NPDU		
	Test Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	

10.7.2 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 - 14.6.X1 - Holds at Least 5 Foreign Device Registrations		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 14.6.X2 - Negative Foreign Device Registration when FD_Supported is FALSE		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
135.1-2013 - 14.6.1 - Execute Read-Foreign-Device-Table		
	Test Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	
135.1-2013 - 14.6.3.1 - Non-zero-Duration Foreign Device Table Timer Operations		
	Test Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	
135.1-2013 - 14.6.5 - Execute Delete-Foreign-Device-Table-Entry Which Should Be Rejected		
	Test Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	

135.1-2013 - 14.6.6 - Execute Delete-Foreign-Device-Table-Entry		
	Test Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	
BTL - 14.7.3.2 - Broadcast Message From a Foreign Device (Two-hop Distribution)		
	Test Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	

10.7.3 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 Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	
135.1-2013 - 14.3.2 - Execute Write-Broadcast-Distribution-Table (Table Shrinkage)		
	Test Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	
BTL - 14.3.3 - Verify Broadcast Distribution Table Created from the Configuration Saved During the Previous Session		
	Test Conditionality	This test may be skipped if the IUT claims support for BACnet/IP - BBMD Functionality.
	Test Directives	
	Testing Hints	
BTL - 14.3.X2 - Broadcast_Distribution_Table Holds at Least 5 Entries (via Write-Broadcast-Distribution-Table)		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

10.7.4 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 - 14.3.X3 - BBMD_Broadcast_Distribution_Table Holds at Least 5 Entries (via BBMD_Broadcast_Distribution_Table)		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL – 14.3.X1 - Write-BDT service is required to return Write-BDT-NAK		
	Test Conditionality	Must be executed in all devices claiming Protocol_Revision >= 17.
	Test Directives	
	Testing Hints	

13.5 Audit Reporting - View - A

Addendum 135-2016*bi* added Audit Reporting. This section adds support to the BTL Test Package for claiming AR-V-A.

These changes are not contained in any SSPC proposal.

Checklist Changes

[In BTL Checklist, replace Audit Reporting sections]

Audit Reporting - View - A		
	R+	Base Requirements
	C[1]	Supports initiation of AuditLogQuery by Target
	C[1]	Supports initiation of AuditLogQuery by Source
	C[1]	Supports initiation of ReadRange
	[1] At least one of these must be supported.	
Audit Reporting - Advanced View and Modify - A		
	R+	Base Requirements
	C[1]	Supports initiation of AuditLogQuery by Target
	C[1]	Supports initiation of AuditLogQuery by Source
	[1] At least one of these must be supported.	

Test Plan Changes

[In BTL Test Plan, replace section 13.5 Audit Reporting-View-A]

13.5 Audit Reporting-View-A

13.5.1 Base Requirements

Base requirements must be met by any IUT that supports AR-V-A.

13.5.2 Supports Initiation of AuditLogQuery By Target

BTL - 8.X.2 - Query and Present Audit Log Records By Target		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

13.5.3 Supports Initiation of AuditLogQuery By Source

BTL - 8.X.1 - Query and Present Audit Log Records By Source		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

13.5.4 Supports Initiation of ReadRange

BTL - 8.18.X1 - Reading and Presenting Large List Properties		
	Test Conditionality	Must be executed.
	Test Directives	Apply on Log_Buffer property of an AuditLog and verify that each record is completely presented.
	Testing Hints	

13.6 Audit Reporting-Advanced View and Modify-A

13.6.1 Base Requirements

Base requirements must be met by any IUT that supports AR-AVM-A.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims AR-V-A
	Testing Hints	
Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims DS-RP-A
	Testing Hints	
Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims DS-WP-A
	Testing Hints	
Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims DM-OCD-A and is able to create Audit Reporter and Audit Log objects.
	Testing Hints	
135.1-2013 - 8.18.3 - Reading and Presenting Properties		
	Test Conditionality	Must be executed.
	Test Directives	Repeat for all properties of the Audit Reporter and Audit Log objects specified by AR-AVM-A, and for all audit related properties in a randomly chosen set of other standard object types.
	Testing Hints	
135.1-2013 - 8.22.4 - Accepting Input and Modifying Properties		
	Test Conditionality	Must be executed.
	Test Directives	Repeat for all properties of the Audit Reporter and Audit Log objects specified by AR-AVM-A, and for all audit related properties in a randomly chosen set of other standard object types.
	Testing Hints	

13.5.2 Supports Initiation of AuditLogQuery By Target

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AuditLogQuery by Target for AR-V-A.
	Testing Hints	

13.5.3 Supports Initiation of AuditLogQuery By Source

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for AuditLogQuery by Source for AR-V-A.
	Testing Hints	

BTL Specified Tests Changes

This section contains all of the new and changed tests required by the interim test BTL Checklist and BTL Test Plan changes.

[Network Port Object Tests]
[In BTL Specified Tests, add clause 7.3.2.X43 Network Port Object Tests]

7.3.2.X43 Network Port Object Tests

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)
```

[Elevator Group, Escalaror, and Lift Object Tests]
[In BTL Specified Tests, add clause 7.3.2.X45]

7.3.2.X45 Elevator Group Object Tests

7.3.2.X45.1 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.X45.2 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.X45.3 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.X45.4 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.

[In BTL Specified Tests, add clause 7.3.2.X46]

7.3.2.X46 Escalator Object Tests

7.3.2.X46.1 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.X46.2 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.X46.3 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.X46.4 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.X46.5 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.X46.6 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.X46.7 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.X46.8 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.X46.7 except that the ConfirmedEventNotification requests are UnconfirmedEventNotification requests and the TD does not acknowledge receiving the notifications.

[Elevator Group, Escalator, and Lift Object Tests]
[In BTL Specified Tests, add clause 7.3.2.X47 Lift Object Tests]

7.3.2.X47 Lift Object Tests

7.3.2.X47.1 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.X47.2 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.X47.3 Car_Door_Status and Landing_Door_Status Tracking Test

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

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

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

Test Steps:

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

7.3.2.X47.4 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.X47.5 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.X47.6 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.X47.7 Assigned_Landing_Call and Registered_Car_Call Tracking Test

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

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

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

Test Steps:

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

7.3.2.X47.8 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.X47.9 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.X47.10 Making_Car_Call and Registered_Car_Call Test

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

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

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

Test Steps:

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

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

7.3.2.X47.11 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.X47.12 Landing_Door_Status Tracks Car_Door_Status Test

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

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

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

Test Steps:

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

7.3.2.X47.13 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.X47.14 Highest Universal floor number linking to Assigned_Landing_Calls, Making_Car_Call and Registered_Car_Call properties

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

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

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

Test Steps:

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

7.3.2.X47.15 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.X47.16 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.X47.17 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.

[Elevator Group, Escalator, and Lift Object Tests]
[In BTL Specified Tests, add to Clause 8.4.X9]

8.4.X9 CHANGE_OF_RELIABILITY Tests

8.4.X9.13 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

[Elevator Group, Escalator, and Lift Object Tests]
[In BTL Specified Tests, add to Clause 8.5.X9]

8.5.X9 CHANGE_OF_RELIABILITY Tests

8.5.X9.14 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.

8.5.X9.15 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
```

[Audit Reporting Tests -- copied from TP 18 Slave Proxy tests in Addendum 16.1-misc1]
[Add clause 8.18.X1 into BTL Specified Tests]

8.18 ReadRange Service Initiation Tests

8.18.X1 Reading and Presenting Large List Properties
Reason for Change: there is no appropriate test for reading and presenting large list property values as required by DM-SP-VM-A.

Purpose: This test case verifies that the IUT is capable of reading and presenting large list properties using ReadRange. It is a generic test used to test data presentation requirements.

Configuration: For this test, the tester shall choose a list property, P1, from an object, O1. The TD shall be configured to not support segmentation. The value is P1 shall be too large to read via ReadProperty or ReadPropertyMultiple.

Test Steps:

```
1.  MAKE (the IUT read P1)
2.  WHILE (the complete list has not been read)
        RECEIVE ReadRange-Request,
            'Object Identifier' =               O1,
            'Property Identifier' =     P1,
            'Range' =                           (any valid value for P1)
        TRANSMIT BACnet-ComplexACK-PDU,
            'Object Identifier'    =            O1,
            'Property Identifier' =     P1,
            'Result Flags' =                    (values consistent with the request),
            'Item Count' =                      (values consistent with the request),
            'Item Data' =               (values consistent with the request)
3.  CHECK (that the IUT presents a list of values that is consistent with the values received in step 2)
```

Notes to Tester: The value presented by the IUT may differ from the value transmitted on the wire due to rounding, truncation, formatting, language conversion, etc.

Notes to Tester: If the IUT has not already determined that the value cannot be read using ReadProperty or ReadPropertyMultiple, the IUT may initiate a ReadProperty or ReadPropertyMultiple. If this occurs, the IUT shall pass the test only if it automatically falls back to using ReadRange upon receipt of the correct BACnetReject-PDU from the TD, indicating that the response is too large.

[Audit Reporting Tests]
[Insert clause 8.X]

8.X AuditLogQuery Initiation Tests

This clause defines the tests necessary to demonstrate support for initiating AuditLogQuery service requests.

8.X.1 Query and Present Audit Log Records By Source
Reason for Change: no tests exist for the functionality.

Purpose: To verify that the IUT correctly initiates AuditLogQuery requests and presents the results.

Test Concept: The TD is setup with an AuditLog containing content from multiple sources and for multiple targets. The audit log contains example entries of all possible operations (see clause 19.Y.5) for audit source S1 with a mix of success and failure entries. The IUT is made to request and display the contents of the Audit Log for source S1. The results are verified that they match the content of the log.

Test Configuration:

<move the log description here>

Test Steps:

```
1.  WHILE (the IUT has not retrieved and displayed all entries for S1)
        MAKE (the IUT request more content from the Audit Log)
        RECEIVE AuditLogQuery-Request
            'Audit Log' =                   (the audit log in the TD),
            'Query Parameters' =            (a valid Audit Query by Source query including S1 as
                                            the source),
            'Start At Sequence Number' =    (any valid value)
            'Requested Count' =             (any valid value)
        TRANSMIT AuditLogQuery-Result
            'Audit Log' =                   (the audit log in the TD),
            'Records' =                     (the set of audit log records which match the query
                                            and which fit within the accepted response size),
            'No More Items' =               (TRUE if the last item is included, FALSE otherwise)
2.  CHECK(that the displayed content matches audit records returned and that the complete records are
        presented)
```

Notes to Tester: If manual interaction is required between subsequent AuditLogQuery requests, checking of the displayed content might need to be performed before the manual interaction is taken instead of at the end of retrieving all of the items.

8.X.2 Query and Present Audit Log Records By Target
Reason for Change: no tests exist for the functionality.

Purpose: To verify that the IUT correctly initiates AuditLogQuery requests and presents the results.

Test Concept: TD is setup as an audit logger with an Audit Log. The IUT is made to request and display the contents of the Audit Log for target T1. The results are verified that they match the content of the log.

Test Configuration: The TD is setup with an AuditLog containing content from multiple sources and for multiple targets. The audit log contains example entries of all possible operations (see clause 19.Y.5) for audit target T1 with a mix of success and failure entries.

Test Steps:

```
1.  WHILE (the IUT has not retrieved and displayed all entries for T1)
        MAKE (the IUT request more content from the Audit Log)
        RECEIVE AuditLogQuery-Request
            'Audit Log' =                   (the audit log in the TD),
            'Query Parameters' =            (a valid Audit Query by Target query including T1 as
                                            the target),
            'Start At Sequence Number' =    (any valid value)
            'Requested Count' =             (any valid value)
        TRANSMIT AuditLogQuery-Result
            'Audit Log' =                   (the audit log in the TD),
            'Records' =                     (the set of audit log records which match the query
```

and which fit within the accepted response size),
'No More Items' = (TRUE if the last item is included, FALSE otherwise)

2. CHECK(that the displayed content matches audit records returned and that the complete records are presented)

Notes to Tester: If manual interaction is required between subsequent AuditLogQuery requests, checking of the displayed content might need to be performed before the manual interaction is taken instead of at the end of retrieving all of the items.

[WriteGroup Tests]
[In BTL Specified Tests, add clause 8.X WriteGroup Service Initiation Tests]

8.X2 WriteGroup Service Initiation Tests

This clause defines the tests necessary to demonstrate support for initiating WriteGroup service requests.

BACnet Reference Clause: 15.11.

8.X2.1 Broadcasting to a Group of Channel Objects

Purpose: Verify that the IUT can initiate a WriteGroup request to an arbitrary group with an arbitrary channel number.

Test Concept: Make the IUT perform a WriteGroup request to a tester selected group and channel. Verify that the request is generated.

Test Steps:

1. MAKE(the IUT initiate a WriteGroup request to group G and Channel C)
2. RECEIVE WriteGroup-Request

DESTINATION =	GLOBAL_BROADCAST | LOCAL_BROADCAST | REMOTE_BROADCAST | TD,
'Group Number' =	G,
'Write Priority' =	(any valid value),
'Change List' =	(a valid list of 1 or more changes which impact channel C),
'Inhibit Delay' =	(absent or TRUE or FALSE)

[Network Port Object Tests]
[Add into clause 9.18.1]

9.18 ReadProperty Service Execution Tests

9.18.1 Positive ReadProperty Service Execution Tests

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.

[BACnet/IPv6 Tests]
[Insert new Clause 12.X, p. 643]

12.X. BACnet/IPv6 Functionality Tests

This clause defines the tests necessary to demonstrate BACnet/IPv6 functionality, as defined in Annex U of the BACnet Standard. For each test case a sequence of one or more messages that are to be exchanged are described. A passing result occurs when the IUT and TD exchange messages as described in the test case. Any other combinations of messages constitute a failure of the test. Some test cases are not valid unless some other test defined in this standard has already been executed and the IUT passed this test. These dependencies are noted in the test case description.

For the tests in this clause references to the virtual address mean the 3-octet virtual address. For example, Source-Virtual-Address = TD means Source-Virtual-Address = (the 3-octet VMAC of TD).

12.X.1 Common Tests

This group of tests verifies that a B/IPv6 device will respond correctly to incoming B/IPv6 messages. All B/IPv6 devices shall execute these tests.

Configuration Requirements: The IUT's Network Port object that represents the B/IPv6 port under test shall be configured as follows:

- BACnet_IPv6_Multicast_Address is FF02::BAC0 (Link Local Multicast Address)

12.X.1.1 Execute Original-Unicast-NPDU

Purpose: To verify that an IUT will process an Original-Unicast-NPDU message.

Test Steps:

1. TRANSMIT DA = IUT, SA = TD,
   Original-Unicast-NPDU,
   Source-Virtual-Address = TD,
   Destination-Virtual-Address = IUT,
   ReadProperty-Request,
      'Object Identifier' = X,
      'Property Identifier' = Y
2. RECEIVE DA = TD, SA= IUT

```
        Original-Unicast-NPDU,
        Source-Virtual-Address = IUT,
        Destination-Virtual-Address = TD,
        ReadProperty-ACK,
                'Object Identifier' = X,
                'Property Identifier' = Y
```

12.X.1.2 Execute Virtual-Address-Resolution
Purpose: To verify that an IUT will process a Virtual-Address-Resolution message.

Test Steps:

```
1. TRANSMIT DA = IUT, SA = TD,
        Virtual-Address-Resolution,
        Source-Virtual-Address = TD
2. RECEIVE DA = TD,
        Virtual-Address-Resolution-ACK,
        Source-Virtual-Address = IUT,
        Destination-Virtual-Address = TD
```

12.X.2 IPv6 Normal Mode Tests
This group of tests verifies that a B/IPv6 device that is operating in normal mode (not a BACnet Broadcast Management Device (BBMD), and not a Foreign Device) will respond correctly to incoming B/IPv6 messages.

Configuration Requirements: The IUT's Network Port object that represents the B/IPv6 port under test shall be configured as follows:

- BACnet_IPv6_Mode is NORMAL
- BACnet_IPv6_Multicast_Address is FF02::BAC0 (Link Local Multicast Address)

12.X.2.1 Positive Tests

12.X.2.1.1 Initiate Original-Broadcast-NPDU
Purpose: To verify that an IUT, operating in normal IPv6 mode, will correctly initiate an Original-Broadcast-NPDU message.

Test Steps:

```
1. MAKE(the IUT send a broadcast)
2. RECEIVE DA=Link Local Multicast Address, SA = IUT
        Original-Broadcast-NPDU,
        Source-Virtual-Address = IUT,
        (any valid BACnet-Unconfirmed-Request-PDU, with any valid broadcast network options)
```

12.X.2.1.2 Execute Original-Broadcast-NPDU
Purpose: To verify that an IUT, operating in normal IPv6 mode, will process an Original-Broadcast-NPDU message.

Test Steps:

```
1. TRANSMIT DA = B/IPv6 Link Local Multicast Address, SA = TD,
        Original-Broadcast-NPDU,
        Source-Virtual-Address = TD,
        Who-Is-Request
2. IF (the IUT responds with Unicast I-Am) THEN
        RECEIVE DA = TD, SA = IUT,
                Original-Unicast-NPDU,
                Source-Virtual-Address = IUT,
                Destination-Virtual-Address = TD,
                I-Am-Request
    ELSE
        RECEIVE DA=Link Local Multicast Address, SA = IUT
                Original-Broadcast-NPDU,
                Source-Virtual-Address = IUT,
                I-Am-Request
3. CHECK (The IUT does not issue any Forwarded-NPDUs)
```

12.X.2.1.3 Execute Forwarded-NPDU

Purpose: To verify that an IUT, operating in normal IPv6 mode, will process a Forwarded-NPDU.

Test Steps:

```
1. TRANSMIT DA = Link Local Multicast Address, SA = TD,
        Forwarded-NPDU,
        Original-Source-Virtual-Address = D2,
        Original-Source-B/IPv6-Address = D2,
        Who-Is-Request
2. IF (the IUT responds with Unicast I-Am) THEN
        RECEIVE DA = D2, SA = IUT,
                Original-Unicast-NPDU,
                Source-Virtual-Address = IUT,
                Destination-Virtual-Address = D2,
                I-Am-Request
   ELSE
        RECEIVE DA=Link Local Multicast Address, SA = IUT
                Original-Broadcast-NPDU,
                Source-Virtual-Address = IUT,
                I-Am-Request
3. CHECK (The IUT does not issue any Forwarded-NPDU BVLCs)
```

12.X.2.1.4 Execute Address-Resolution

Purpose: To verify that an IUT, operating in normal IPv6 mode, will process an Address-Resolution message.

Test Steps:

```
1. TRANSMIT DA = B/IPv6 Link Local Multicast Address, SA = TD,
        Address-Resolution,
        Source-Virtual-Address = TD,
        Target-Virtual-Address = IUT
2. RECEIVE DA = TD,
        Address-Resolution-ACK,
        Source-Virtual-Address = IUT,
        Destination-Virtual-Address = TD
3. CHECK (The IUT does not issue any Forwarded-Address-Resolution BVLCs)
```

12.X.2.1.5 Execute Forwarded-Address-Resolution

Purpose: To verify that an IUT, operating in normal IPv6 mode, will process a Forwarded-Address-Resolution message.

Test Concept: The TD, acting as a BBMD, sends a Forwarded-Address-Resolution message to the IUT on behalf of device D2. It is verified that the IUT responds to D2 with an Address-Resolution message.

```
1. TRANSMIT DA = IUT, SA = TD,
        Forwarded-Address-Resolution,
```

Original-Source-Virtual-Address = D2,
Target-Virtual-Address = IUT
Original-Source-B/IPv6-Address = D2

2. RECEIVE
DA = D2, SA = IUT
Address-Resolution-ACK,
Source-Virtual-Address = IUT,
Destination-Virtual-Address = D2

3. CHECK (The IUT does not issue any Forwarded-Address-Resolution BVLCs).

12.X.2.2 Negative Tests

12.X.2.2.1 Reject Register-Foreign-Device

Purpose: To verify that an IUT, operating in normal IPv6 mode, will reject a Register-Foreign-Device request.

Test Steps:

1. TRANSMIT DESTINATION = IUT, SA = TD,
Register-Foreign-Device,
Source-Virtual-Address = TD
Time-To-Live = 60
2. RECEIVE DESTINATION = TD,
BVLC-Result,
Source-Virtual-Address = IUT
'Result Code' = Register-Foreign-Device NAK

12.X.2.2.2 Reject Delete-Foreign-Device-Table-Entry

Purpose: To verify that an IUT, operating in normal IPv6 mode, will reject a Delete-Foreign-Device-Table-Entry request.

Test Steps:

1. TRANSMIT DESTINATION = IUT, SA = TD,
Delete-Foreign-Device-Table-Entry,
Source-Virtual-Address = TD
FDT Entry = TD
2. RECEIVE DESTINATION = TD,
BVLC-Result,
Source-Virtual-Address = IUT
'Result Code' = Delete-Foreign-Device-Table-Entry NAK

12.X.2.2.3 Reject Distribute-Broadcast-To-Network

Purpose: To verify that an IUT, operating in normal IPv6 mode, will reject a Distribute-Broadcast-To-Network request.

Test Steps:

1. TRANSMIT DESTINATION = IUT, SA = TD,
Distribute-Broadcast-To-Network,
Original-Source-Virtual-Address = TD
Who-Is-Request
2. RECEIVE DESTINATION = TD,
BVLC-Result,
Source-Virtual-Address = IUT
'Result Code' = Distribute-Broadcast-To-Network NAK

12.X.3 Foreign Device Tests

This group of tests verifies that a B/IPv6 device that is configured as a Foreign Device is able to register with a BBMD and send and receive broadcast messages through the BBMD.

Configuration Requirements: The IUT's Network Port object that represents the B/IPv6 port under test shall be configured as follows:

- BACnet_IPv6_Mode is FOREIGN
- BACnet_IPv6_Multicast_Address is FF02::BAC0 (Link Local Multicast Address)

12.X.3.1 Positive Tests

12.X.3.1.1 Initiate Distribute-Broadcast-To-Network-NPDU

Purpose: To verify that an IUT, configured as a Foreign Device, will correctly initiate an Distribute-Broadcast-To-Network -NPDU message.

Configuration Requirements: The TD is operating as a BBMD, and the IUT has registered as a foreign device with it.

Test Steps:

```
1. MAKE(the IUT send a broadcast)
2. RECEIVE DA=IUT, SA = IUT
        Distribute-Broadcast-To-Network-NPDU,
        Source-Virtual-Address = IUT,
        (any valid BACnet-Unconfirmed-Request-PDU, with any valid broadcast network options)
```

12.X.3.1.2 Execute Forwarded-NPDU

Purpose: To verify that an IUT, operating as a foreign device, will process a Forwarded-NPDU.

Configuration Requirements: The TD is operating as a BBMD, and the IUT has registered as a foreign device with it.

Test Steps:

```
1. TRANSMIT DA = IUT, SA = TD,
        Forwarded-NPDU,
        Original-Source-Virtual-Address = D2,
        Original-Source-B/IPv6-Address = D2,
        Who-Is-Request
2. If (the IUT responds with Unicast I-Am) THEN
        RECEIVE DA = D2, SA = IUT,
                Original-Unicast-NPDU,
                Source-Virtual-Address = IUT,
                Destination-Virtual-Address = D2,
                I-Am-Request
   ELSE
        RECEIVE DA=TD, SA = IUT
                Distribute-Broadcast-To-Network-NPDU,
                Source-Virtual-Address = IUT,
                I-Am-Request
3. CHECK (The IUT does not issue any Forwarded-NPDU BVLCs)
```

12.X.3.1.3 Execute Forwarded-Address-Resolution

Purpose: To verify that an IUT, operating as a foreign device, will process a Forwarded-Address-Resolution message.

Test Concept: The TD, acting as a BBMD, sends a Forwarded-Address-Resolution message to the IUT on behalf of device D2. It is verified that the IUT responds to D2 with an Address-Resolution message.

1. TRANSMIT DA = IUT, SA = TD,
    Forwarded-Address-Resolution,
    Original-Source-Virtual-Address = D2,
    Target-Virtual-Address = IUT
    Original-Source-B/IPv6-Address = D2
2. RECEIVE
    DA = D2, SA = IUT
    Address-Resolution-ACK,
    Source-Virtual-Address = IUT,
    Destination-Virtual-Address = D2
3. CHECK (The IUT does not issue any Forwarded-Address-Resolution BVLCs)

12.X.3.2 Negative Tests

12.X.3.2.1 Ignores Original-Broadcast-NPDU

Purpose: To verify that an IUT, operating as a foreign device, will not process an Original-Broadcast-NPDU message.

Test Steps:

1. TRANSMIT DA = B/IPv6 Link Local Multicast Address, SA = D2,
    Original-Broadcast-NPDU,
    Source-Virtual-Address = D2,
    Who-Is-Request
2. CHECK (The IUT does not issue any IAms in response)

12.X.3.2.2 Ignore Address-Resolution

Purpose: To verify that an IUT, operating as a foreign device, will ignore multicast Address-Resolution messages.

Test Steps:

1. TRANSMIT DA = B/IPv6 Link Local Multicast Address, SA = D2,
    Address-Resolution,
    Source-Virtual-Address = D2,
    Target-Virtual-Address = IUT
2. CHECK (The IUT does not issue any Address-Resolution-ACK BVLCs)

12.X.3.2.3 Reject Register-Foreign-Device

Purpose: To verify that an IUT, operating as a foreign device, will reject a Register-Foreign-Device request.

Test Steps:

1. TRANSMIT DESTINATION = IUT, SA = TD,
    Register-Foreign-Device,
    Source-Virtual-Address = TD
    Time-To-Live = 60
2. RECEIVE DESTINATION = TD,
    BVLC-Result,
    Source-Virtual-Address = IUT
    'Result Code' = Register-Foreign-Device NAK

12.X.3.2.4 Reject Delete-Foreign-Device-Table-Entry

Purpose: To verify that an IUT, operating as a foreign device, will reject a Delete-Foreign-Device-Table-Entry request.

Test Steps:

1. TRANSMIT DESTINATION = IUT, SA = TD,
   Delete-Foreign-Device-Table-Entry,
   Source-Virtual-Address = TD
   FDT Entry = TD
2. RECEIVE DESTINATION = TD,
   BVLC-Result,
   Source-Virtual-Address = IUT
   'Result Code' = Delete-Foreign-Device-Table-Entry NAK

12.X.3.2.5 Reject Distribute-Broadcast-To-Network

Purpose: To verify that an IUT, operating as a foreign device, will reject a Distribute-Broadcast-To-Network request.

Test Steps:

1. TRANSMIT DESTINATION = IUT, SA = TD,
   Distribute-Broadcast-To-Network,
   Original-Source-Virtual-Address = TD
   Who-Is-Request
2. RECEIVE DESTINATION = TD,
   BVLC-Result,
   Source-Virtual-Address = IUT
   'Result Code' = Distribute-Broadcast-To-Network NAK

12.X.4 BBMD Tests

12.X.4.1 Positive Tests

This group of tests verifies that a B/IPv6 device that is configured as a BACnet Broadcast Management Device (BBMD) will correctly process incoming B/IPv6 messages that pertain to BBMDs. Only devices that are configured to support BBMD functionality shall execute these tests.

Configuration Requirements: The IUT's Network Port object that represents the B/IPv6 port under test shall be configured as follows:

- BACnet_IPv6_Mode is BBMD
- BACnet_IPv6_Multicast_Address is FF02::BAC0 (Link Local Multicast Address)
- BBMD_Broadcast_Distribution_Table shall contain:

bbmd-address
BBMD1
BBMD2
BBMD3

For purposes of these tests, TD shall be operating as BBMD1.

12.X.4.1.1 Original-Broadcast-NPDU

Purpose: To verify that the IUT, configured as a BBMD, will forward an Original-Broadcast-NPDU request.

Test Steps:

1. TRANSMIT

DA = B/IPv6 Link Local Multicast Address,
SA = TD,
Source-Virtual-Address = TD,
Original-Broadcast-NPDU,
Who-Is-Request

2. RECEIVE
DA = BBMD1,
SA = IUT,
Forwarded-NPDU,
Original-Source-Virtual-Address = TD
Original-Source-B/IPv6-Address = TD
Who-Is-Request

3. RECEIVE
DA = BBMD2,
SA = IUT,
Forwarded-NPDU,
Original-Source-Virtual-Address = TD
Original-Source-B/IPv6-Address = TD
Who-Is-Request

4. RECEIVE
DA = BBMD3,
SA = IUT,
Forwarded-NPDU,
Original-Source-Virtual-Address = TD
Original-Source-B/IPv6-Address = TD
Who-Is-Request

12.X.4.1.2 Forwarded-NPDU

Purpose: To verify that the IUT, configured as a BBMD, will forward a Forwarded-NPDU request.
Configuration Requirements: Register FD1 as a foreign device with the IUT. FD3 is a registered foreign device with BBMD1.

Test Steps:

1. TRANSMIT
DA = IUT,
SA = BBMD1,
Forwarded-NPDU,
Source-Virtual-Address = FD3,
Original-Source-B/IPv6-Address = FD3
I-Am-Request

2. RECEIVE
DA = B/IPv6 Link Local Multicast Address,
SA = IUT
Forwarded-NPDU,
Source-Virtual-Address = FD3,
Original-Source-B/IPv6-Address = FD3
I-Am-Request

3. RECEIVE
DA = FD1,
SA = IUT
Forwarded-NPDU,
Source-Virtual-Address = FD3,
Original-Source-B/IPv6-Address = FD3
I-Am-Request

Notes to Tester: The order of the messages transmitted by the IUT is not significant.

12.X.4.1.3 Address-Resolution

Purpose: To verify that the IUT, configured as a BBMD, will process an Address-Resolution request when the target virtual address is not the virtual address of the IUT.

Configuration Requirements: TD shall be a registered foreign device (FD1) with the IUT.

1. TRANSMIT
    DA = IUT,
    SA = TD,
    Address-Resolution,
    Source-Virtual-Address = TD,
    Target-Virtual-Address = FD2
2. RECEIVE
    DA = B/IPv6 Link Local Multicast Address
    SA = IUT,
    Forwarded-Address-Resolution,
    Original-Source-Virtual-Address = TD,
    Target-Virtual-Address = FD2,
    Original-Source-B/IPv6-Address = TD
3. RECEIVE
    DA = BBMD1,
    SA = IUT,
    Forwarded-Address-Resolution,
    Original-Source-Virtual-Address = TD,
    Target-Virtual-Address = FD2,
    Original-Source-B/IPv6-Address = TD
4. RECEIVE
    DA = BBMD2,
    SA = IUT,
    Forwarded-Address-Resolution,
    Original-Source-Virtual-Address = TD,
    Target-Virtual-Address = FD2,
    Original-Source-B/IPv6-Address = TD
5. RECEIVE
    DA = BBMD3,
    SA = IUT,
    Forwarded-Address-Resolution,
    Original-Source-Virtual-Address = TD,
    Target-Virtual-Address = FD2,
    Original-Source-B/IPv6-Address = TD
6. RECEIVE
    DA = FD2,
    SA = IUT,
    Forwarded-Address-Resolution,
    Original-Source-Virtual-Address = TD,
    Target-Virtual-Address = FD2,
    Original-Source-B/IPv6-Address = TD
7. TRANSMIT
    DA = TD,
    SA = FD2,
    Address-Resolution-ACK,
    Source-Virtual-Address = FD2,
    Destination-Virtual-Address = TD

Notes to Tester: The execution of step 7 is not significant, but is shown here in order to demonstrate the completion of the BVLC.

12.X.4.1.4 Forwarded-Address-Resolution

Purpose: To verify that the IUT, configured as a BBMD, will process a Forwarded-Address-Resolution request when the target virtual address is not the virtual address of the IUT.

Configuration Requirements: TD shall operate as BBMD1 and listed in the IUTs Broadcast Distribution Table.

1. TRANSMIT
    DA = IUT,
    SA = TD,
    Forwarded-Address-Resolution,
    Original-Source-Virtual-Address = FD1,
    Target-Virtual-Address = FD2
    Original-Source-B/IPv6-Address = FD1
2. RECEIVE
    DA = B/IPv6 Link Local Multicast Address,
    SA = IUT,
    Forwarded-Address-Resolution,
    Original-Source-Virtual-Address = FD1,
    Target-Virtual-Address = FD2,
    Original-Source-B/IPv6-Address = FD1
3. RECEIVE
    DA = FD2,
    SA = IUT,
    Forwarded-Address-Resolution,
    Original-Source-Virtual-Address = FD1,
    Target-Virtual-Address = FD2,
    Original-Source-B/IPv6-Address = FD1
4. TRANSMIT
    DA = TD,
    SA = FD2,
    Address-Resolution-ACK,
    Source-Virtual-Address = FD2,
    Destination-Virtual-Address = TD

Notes to Tester: The execution of step 7 is not significant, but is shown here in order to demonstrate the completion of the BVLC. The order of the messages transmitted by the IUT is not significant.

12.X.4.1.5 Distribute-Broadcast-To-Network

Purpose: To verify that the IUT, configured as a BBMD, will process a Distribute-Broadcast-To-Network request.

Configuration Requirements: Register FD1 as a foreign device with the IUT. FD2 is a registered foreign device with BBMD1. For purposes of this test, TD is acting as FD1.

Steps 1 6 are the processing of the Distributed-Broadcast-To-Network, Step 7 and on is the processing of the APDU service by the IUT.

1. TRANSMIT
    DA = IUT,
    SA = FD1,
    Distribute-Broadcast-To-Network,
    Who-Is-Request
2. RECEIVE
    DA = B/IPv6 Link Local Multicast Address,

SA = IUT,
Forwarded-NPDU,
Source-Virtual-Address = FD1,
Original-Source-Virtual-Address = FD1,
Who-Is-Request

3. RECEIVE
    DA = BBMD1,
    SA = IUT,
    Forwarded-NPDU,
    Source-Virtual-Address = FD1,
    Original-Source-Virtual-Address = FD1,
    Who-Is-Request
4. RECEIVE
    DA = BBMD2,
    SA = IUT,
    Forwarded-NPDU,
    Source-Virtual-Address = FD1,
    Original-Source-Virtual-Address = FD1,
    Who-Is-Request
5. RECEIVE
    DA = BBMD3,
    SA = IUT,
    Forwarded-NPDU,
    Source-Virtual-Address = FD1,
    Original-Source-Virtual-Address = FD1,
    Who-Is-Request
6. RECEIVE
    DA = FD2,
    SA = IUT,
    Forwarded-NPDU,
    Source-Virtual-Address = FD1,
    Original-Source-Virtual-Address = FD1,
    Who-Is-Request
7. RECEIVE
    DA = B/IPv6 Link Local Multicast Address,
    SA = IUT,
    Original-Broadcast-NPDU,
    Original-Source-Virtual-Address = IUT,
    I-Am-Request
8. RECEIVE
    DA = BBMD1,
    SA = IUT,
    Forwarded-NPDU,
    Source-Virtual-Address = IUT,
    Original-Source-Virtual-Address = IUT,
    I-Am-Request
9. RECEIVE DA = BBMD2,
    SA = IUT,
    Forwarded-NPDU,
    Source-Virtual-Address = IUT,
    Original-Source-Virtual-Address = IUT,
    I-Am-Request
10. RECEIVE DA = BBMD3,
    SA = IUT,
    Forwarded-NPDU,
    Source-Virtual-Address = IUT,

    Original-Source-Virtual-Address = IUT,
    I-Am-Request
11. RECEIVE
    DA = FD1,
    SA = IUT,
    Forwarded-NPDU,
    Source-Virtual-Address = IUT,
    Original-Source-Virtual-Address = IUT,
    I-Am-Request
12. RECEIVE
    DA = FD2,
    SA = IUT
    Forwarded-NPDU,
    Source-Virtual-Address = IUT,
    Original-Source-Virtual-Address = IUT,
    I-Am-Request

Notes to Tester: The order of the messages transmitted by the IUT is not significant.

12.X.4.2 Negative Tests

12.X.4.2.1 Reject Forwarded-NPDU

Purpose: To verify that the IUT, configured as a BBMD, will drop a Forwarded-NPDU request from a BBMD that's not in the IUT's BDT.

Configuration Requirements: Empty the IUT's BDT. FD3 is a foreign device registered with the IUT.

Test Steps:

1. TRANSMIT
    DA = IUT,
    SA = BBMD1,
    Forwarded-NPDU,
    Source-Virtual-Address = FD3,
    Original-Source-B/IPv6-Address = FD3
    I-Am-Request
2. CHECK (The IUT does not issue any Forwarded-NPDU BVLCs)

12.X.4.2.2 Reject Address-Resolution

Purpose: To verify that the IUT, configured as a BBMD, will not process an Address-Resolution request when the target virtual address is not the virtual address of the IUT and the SA is not from a device registered with the IUT.

Configuration Requirements: TD shall not be a registered foreign device (FD1) with the IUT.

1. TRANSMIT
    DA = IUT,
    SA = TD,
    Address-Resolution,
    Source-Virtual-Address = TD,
    Target-Virtual-Address = FD2
2. RECEIVE
    DA = TD,
    SA = IUT
    BVLC-Result
    Address-Resolution NAK
2. CHECK (The IUT does not issue any Forwarded-Address-Resolution BVLCs)

12.X.4.2.3 Reject Forwarded-Address-Resolution

Purpose: To verify that the IUT, configured as a BBMD, will not process a Forwarded-Address-Resolution request when the source a BBMD that is not present in the IUTs BDT.

Configuration Requirements: Empty the IUT's BDT.

1. TRANSMIT
    DA = IUT,
    SA = TD,
    Forwarded-Address-Resolution,
    Original-Source-Virtual-Address = FD1,
    Target-Virtual-Address = FD2
    Original-Source-B/IPv6-Address = FD1
2. CHECK (The IUT does not issue any Forwarded-Address-Resolution BVLCs)

12.X.4.2.4 Reject Distribute-Broadcast-To-Network

Purpose: To verify that the IUT, configured as a BBMD, will not process a Distribute-Broadcast-To-Network request from a device that is not registered as a foreign device with the IUT.

Configuration Requirements: Ensure the TD is not registered as a foreign device with the IUT and that the TD is not listed in the IUTs FDT.

1. TRANSMIT
    DA = IUT,
    SA = TD,
    Distribute-Broadcast-To-Network,
    Who-Is-Request
2. RECEIVE
    DA = TD
    SA = IUT
    BVLC-Result
    Distribute-Broadcast-To-Network-NAK

12.X.4.3 Broadcast Distribution Table Operations

This group of tests verifies that a BACnet Broadcast Management Device will correctly perform BDT operations.

Configuration Requirements: The IUT's Network Port object that represents the B/IPv6 port under test shall be configured as follows:

- BACnet_IPv6_Mode is BBMD

12.X.4.3.1 Verify writability of the BDT

Purpose: To verify the contents of the broadcast distribution table.

1. TRANSMIT
    WriteProperty-Request,
        'Object Identifier' = (Network Port Object that represents this port),
        'Property Identifier' = BBMD_Broadcast_Distribution_Table
        'Property Value' =(WrittenBDT: a list of valid BACnetBDTEntry)
2. RECEIVE
    BACnetSimple-Ack,
3. READ ReadBDT = NP, BBMD_Broadcast_Distribution_Table
4. CHECK(ReadBDT contains the same entries as WrittenBDT, but not necessarily in the same order)

12.X.5 Foreign Device Management Tests

This group of tests verifies that a BBMD with an FDT will correctly perform FDT operations.

Configuration Requirements: The IUT's Network Port object, NP, that represents the B/IPv6 port under test shall be configured as follows:

- BACnet_IPv6_Mode is BBMD
- BACnet_IPv6_Multicast_Address is FF02::BAC0 (Link Local Multicast Address)
- BBMD_Accept_FD_Registrations is TRUE.

The TD's Network Port object that represents the B/IPv6 port being used shall be configured as follows:

- BACnet_IPv6_Mode is FOREIGN
- BACnet_IPv6_Multicast_Address is FF02::BAC0 (Link Local Multicast Address)

12.X.5.1 Execute Register-Foreign-Device
Purpose: To verify that the IUT, will handle a Register-Foreign-Device request.

Test Steps:

```
1. TRANSMIT
        DA = IUT,
        SA = TD,
        Source-Virtual-Address = TD,
        Register-Foreign-Device,
        'Time-To-Live' = 60
2. RECEIVE
        DA = TD,
        SA = IUT,
        Source-Virtual-Address = IUT,
        BVLC-Result,
        'Result Code' = 0
3. VERIFY NP, BBMD_Foreign_Device_Table = ( (B/IPv6 address of FD2, 60, 90-execution time) )
```

12.X.5.2 Execute Delete-Foreign-Device-Table-Entry
Purpose: To verify that the IUT will handle a Delete-Foreign-Device-Table-Entry message when a valid FDT entry is supplied.

Configuration Requirements: The TD shall take the role of foreign device FD1. The IUT's FDT must be empty.

Test Steps:

```
1. TRANSMIT
        DA = IUT,
        SA = FD1,
        Source-Virtual-Address = FD1,
        Register-Foreign-Device,
        'Time-To-Live' = 60
2. RECEIVE
        DA = FD1,
        SA = IUT,
        Source-Virtual-Address = IUT,
        BVLC-Result,
        'Result Code' = 0
3. VERIFY NP, BBMD_Foreign_Device_Table = ( (B/IPv6 address of FD1, 60, 90-execution time) )
4. TRANSMIT
        DA = IUT,
        SA = FD1,
        Source-Virtual-Address = FD1,
```

Delete-Foreign-Device-Table-Entry,
'FDT Entry' = FD1

5. RECEIVE
    DA = FD1,
    SA = IUT,
    Source-Virtual-Address = IUT,
    BVLC-Result,
    'Result Code' = Successful completion
6. VERIFY NP, BBMD_Foreign_Device_Table = ()

12.X.5.3 Foreign Device Table Timer Operations

12.X.5.3.1 Non-Zero-Duration Foreign Device Table Timer Operations

Purpose: To verify that the IUT will handle FDT timer operations: finite time Foreign Device registration, re-registration, adding grace period to the supplied Time-To-Live parameter and FDT entry clearing upon timer expiration.

Configuration Requirements: The TD shall take the role of foreign device FD2. The value of the IUT's BBMD_Foreign_Device Table must be empty.

Test Steps:

1. TRANSMIT
    DA = IUT,
    SA = FD2,
    Register-Foreign-Device,
    'Time-To-Live' = 60
2. RECEIVE
    DA = FD2,
    SA = IUT,
    BVLC-Result,
    'Result Code' = 0
3. VERIFY NP, BBMD_Foreign_Device_Table = ((B/IPv6 address of FD2, 60, 90-execution time))
4. TRANSMIT
    DA = IUT,
    SA = FD2,
    Register-Foreign-Device,
    'Time-To-Live' = 40
5. RECEIVE
    DA = FD2,
    SA = IUT,
    BVLC-Result,
    'Result Code' = 0
6. WAIT (30 seconds)
7. VERIFY NP, BBMD_Foreign_Device_Table = ((B/IPv6 address of FD2, 40, 40-execution time))
8. WAIT (50 seconds)
9. VERIFY NP, BBMD_Foreign_Device_Table = ( )

12.X.5.3.2 Zero-Duration Foreign Device Timer Operations

Purpose: To verify that the IUT will handle Foreign Device registration with Time-To-Live parameter equal to zero and clears FDT entry upon timer expiration.

Configuration Requirements: The TD shall take the role of foreign device FD2. The IUTs FDT must be empty.

Test Steps:

1. TRANSMIT
    DA = IUT,
    SA = FD2,
    Register-Foreign-Device-Table,
    'Time-To-Live' = 0
2. RECEIVE
    DA = FD2,
    SA = IUT,
    BVLC-Result,
    'Result Code' = 0
3. WAIT (10 seconds)
4. VERIFY NP, BBMD_Foreign_Device_Table = ((B/IPv6 address of FD2, 0, 20-execution time))
5. WAIT (30 seconds)
6. VERIFY NP, BBMD_Foreign_Device_Table = ()

12.X.5.4 Delete-Foreign-Device-Table-Entry For A Non-existent Entry

Purpose: To verify that the IUT will handle a Delete-Foreign-Device-Table-Entry message when a non-existent FDT entry is supplied.

Test Concept: The IUT starts with a Foreign Device Table without a entry for FD1. The TD, acting as FD1, attempts to delete its entry from the IUT's Foreign Device Table. It is verified that the IUT returns a NAK to the request.

Configuration Requirements: The IUT's Foreign Device Table does not contain an entry for FD1.

Test Steps:

1. VERIFY NP, BBMD_Foreign_Device_Table = (FDT1: a list of entries without an entry for FD1)
4. TRANSMIT
    DA = IUT,
    SA = FD1,
    Source-Virtual-Address = FD1,
    Delete-Foreign-Device-Table-Entry,
    'FDT Entry' = FD1
5. RECEIVE
    DA = FD1,
    SA = IUT,
    Source-Virtual-Address = IUT
    BVLC-Result,
    'Result Code' = Delete-Foreign-Device-Table-Entry NAK
6. VERIFY NP, BBMD_Foreign_Device_Table = FDT1

[Network Port Object Tests]
[In BTL Specified Tests, add test into 14.3]
[All other 14.3 Write-BDT tests need to have a conditionality added to them based on the IUT's Protocol_Revision being less than 17]

14.3 Broadcast Distribution Table Operations

14.3.X1 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

[Network Port Object Tests]
[In BTL Specified Tests, add test into 14.3]

14.3.X2 Broadcast Distribution Table Holds at Least 5 Entries (via Write-Broadcast-Distribution-Table)
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: Verify that the Broadcast_Distribution_Table can hold at least five distinct peer BBMDs entries (in addition to the entry containing the address of itself in the table) using Write-Broadcast-Distribution-Table.

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

Test Steps:

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

14.3.X3 Broadcast Distribution Table Holds at Least 5 Entries (via BBMD_Broadcast_Distribution_Table)
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: Verify that the Broadcast_Distribution_Table in the BBMD's Network Port object, NP, can hold at least five distinct peer BBMDs entries (in addition to the entry containing the address of itself in the table) by writing to the BBM_Broadcast_Distribution_Table property.

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

Test Steps:

```
1.	WRITE NP, BBMD_Broadcast_Distribution_Table = (its own entry and entries for at least 5 other BBMDs)
2.	TRANSMIT ReinitializeDevice-Request
		'Reinitialized State of Device' = WARMSTART | ACTIVATE_CHANGES
		'Password' = (any valid password)
3.	RECEIVE BACnet-SimpleACK-PDU
4.	WAIT Activate Change Fail Time
5.	TRANSMIT Read- Broadcast-Distribution-Table
6.	RECEIVE Read-Broadcast-Distribution-Table-Ack,
		'List of BDT Entries' = (the table as configured, in any order)
```

[Network Port Object Tests]
[In BTL Specified Tests, add test into 14.6]

14.6 Foreign Device Management

14.6.X1 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
}
```

14.6.X2 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
```