BACnet® TESTING LABORATORIES
ADDENDA

Addendum h to BTL Test Package 15.2

Revision 6
Revised 3/22/2019

Approved by the BTL Working Group on 10-Jan-2019;
Approved by the BTL Working Group Voting Members on 8-Mar-2019;
Published on 25-Mar-2019

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

FOREWORD

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

In the following document, language to be added to existing clauses within the BTL Test Package 15.2 is indicated through the use of *italics*, while deletions are indicated by ~~strikethrough~~. Where entirely new subclauses are proposed to be added, plain type is used throughout

In addition, changes to BTL Specified Tests also contain a yellow highlight to indicate the changes made by this addendum.

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

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

BTL-15.2h-1: PrivateTransfer Testing [BTLWG-362]

Overview:
This document adds testing for private transfer services.

Changes:
[In BTL Checklist, Add to section 2 Basic BACnet Functionality the following new section.]

Support	Listing	Option
Private Transfer Services		
	R	Base Requirements
	O	Initiates ConfirmedPrivateTransfer
	O	Initiates UnconfirmedPrivateTransfer
	O	Executes ConfirmedPrivateTransfer
	O	Executes UnconfirmedPrivateTransfer

[In BTL Test Plan, Add new section below]

2.3 Private Transfer Services

2.3.1 Base Requirements

Base requirements must be met by any IUT that initiates or executes private transfer services.

2.3.2 Initiates ConfirmedPrivateTransfer

The IUT initiates ConfirmedPrivateTransfer service requests.

135.1-2013 - 8.25 - ConfirmedPrivateTransfer Service Initiation Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

2.3.3 Initiates UnconfirmedPrivateTransfer

The IUT initiates UnconfirmedPrivateTransfer service requests.

135.1-2013 - 8.26 - UnconfirmedPrivateTransfer Service Initiation Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

2.3.4 Executes ConfirmedPrivateTransfer

The IUT initiates ConfirmedPrivateTransfer service requests.

BTL - 9.25.2.1 - Correctly Executes a Non-Supported ConfirmedPrivateTransfer Service		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

2.3.5 Executes UnconfirmedPrivateTransfer

The IUT executes UnconfirmedPrivateTransfer service requests.

There are no tests for this functionality.

[In BTL Specified Tests, Add new section as derived from 135.1-2013 - 9.25]

9.25 ConfirmedPrivateTransfer Service Execution Tests

Reason for change: renumbered so negative tests could be added

This clause defines the tests necessary to demonstrate support for executing ConfirmedPrivateTransfer service requests.

9.25.1 Positive ConfirmedPrivateTransfer Service Execute Tests

9.25.1.*1* Correctly Executes a Supported ConfirmedPrivateTransfer Service
Dependencies: None.

BACnet Reference Clause: 16.2.

Purpose: To verify the ability to correctly execute a ConfirmedPrivateTransfer service request.

Test Concept: The service procedure implied by a particular private transfer service is defined by the vendor. This test simply verifies that an appropriate acknowledgment is returned and that any externally visible actions defined by the vendor are observed.

Configuration Requirements: The IUT shall be configured to execute at least one ConfirmedPrivateTransfer service. The service parameters that are to be provided in the request and a list of any externally visible actions that should be apparent to the tester shall also be provided.

Test Steps:

```
1.  TRANSMIT ConfirmedPrivateTransfer-Request,
        'Vendor ID' =           (the Vendor_Identifier specified in the Device object of the
                    EPICS),
        'Service Number' =      (any service number provided by the vendor),
        'Service Parameters' =  (the service parameters provided for this service)
2.  RECEIVE ConfirmedPrivateTransfer-ACK,
        'Vendor ID' =           (the Vendor_Identifier specified in the Device object of the
                    EPICS),
        'Service Number' =      (the service number used in step 1),
        'Result Block' =        (the expected results provided by the vendor)
3.  CHECK (Did the externally visible actions take place?)
```

[In BTL Specified Tests, Add new test 9.25.2]

<removed test because superseded by changes made in 15.2h-2>

BTL-15.2h-2: PrivateTransfer Testing - revised [BTLWG-475]

Overview:

This document revises the negative testing for ConfirmedPrivateTransfer service, eliminating the specific way that it responds with an error or rejects the service.

Changes:

[In BTL Test Plan, Add new test to section Executes ConfirmedPrivateTransfer in yellow below.]

2.3.4 Executes ConfirmedPrivateTransfer

The IUT executes ConfirmedPrivateTransfer service requests.

BTL - 9.25.1.1 Correctly Executes a Supported ConfirmedPrivateTransfer Service		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 9.25.2.1 Correctly Executes a Non-Supported ConfirmedPrivateTransfer Service		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

[In BTL Specified Tests, add new test]

9.25.2.1 Correctly Executes a Non-Supported ConfirmedPrivateTransfer Service
Reason for Change: There is no existing test verifying correct error responses for unknown ConfirmedPrivateTransfer services.

Purpose: To verify that the IUT correctly responds with an error when an unsupported ConfirmedPrivateTransfer is received.

Test Concept: Send a non-supported ConfirmedPrivateTransfer request to the IUT and verify that it responds with an error or rejects the service.

Test Steps:

```
1.  TRANSMIT ConfirmedPrivateTransfer-Request,
        'Vendor ID' =           (VID: any valid value not supported by the IUT),
        'Service Number' =      (SID: any valid value, that when combined with Vendor ID is not supported by the IUT),
        'Service Parameters' =  (any valid values)
2.  RECEIVE
        (ConfirmedPrivateTransfer-Error,
            'Error Class' =         any
            'Error Code' =          any
            'Vendor ID' =           (VID),
            'Service Number' =      (SID),
            'Result Block' =        (the expected results provided by the vendor)) |
        (BACnet-Reject-PDU,
            'Reject Reason' =       any reason) |
        (BACnet-Abort-PDU
            'Abort Reason' =        any reason)
3.  CHECK (that the IUT exhibits the vendor defined results)
```

BTL-15.2h-3: Life Safety Object Tests [BTLWG-28]

Overview:

Add testing for the Life Safety Object to the Test Package.

Changes:

[In BTL Checklist, add/update the following sections]

Life Safety Point Object		
	R	Base Requirements
	S	Supports writable Out_Of_Service properties
	O	Supports writable Member_Of property
	O	Contains an object with Reliability_Evaluation_Inhibit property
Life Safety Zone Object		
	R	Base Requirements
	S	Supports writable Out_Of_Service properties
	O	Supports writable Member_Of property
	O	Contains an object with Reliability_Evaluation_Inhibit roperty

Alarm and Event - Notification - A

	O	Processes CHANGE_OF_LIFE_SAFETY notifications

Alarm and Event - Life Safety - A

	R	Base Requirements
	R	Initiates LifeSafetyOperation requests
	R	Executes ConfirmedEventNotifications
	R	Executes UnconfirmedEventNotifications
	R	Processes intrinsically generated notifications
	R	Processes algorithmically generated notifications
	R	Processes event notifications with timestamps of the BACnetDateTime form
	R	Processes event notifications with timestamps of the Time form
	R	Processes event notifications with timestamps of the Sequence Number form
	R	Supports AE-ACK-A
	R	Supports AE-INFO-A

Alarm and Event - Life Safety - B

	R	Base Requirements
	R	Supports the Notification Class Object
	R	Supports AE-INFO-B
	C[1]	Implements intrinsic alarming
	C[1]	Supports the Event Enrollment object
	C[2]	Supports AE-ACK-B
	C[3]	Generates event notifications with timestamps of the BACnetDateTime form
	C[3]	Generates event notifications with timestamps of the Time form
	C[3]	Generates event notifications with timestamps of the Sequence Number form
	O	Mode Transition Tests when Event State is Maintained
	O	Supports Event_Message_Texts property
	O	Supports Event_Message_Texts_Config property
[1] At least one of these options must be supported to claim support for this BIBB. [2] Required if EventNotifications with service parameter AckRequired = True can be issued. [3] At least one of these options must be supported to claim support for this BIBB. The BACnetDateTime form of the timestamp is the recommended option.		

[In BTL Test Plan, add new section for Life Safety Point Object as below]

3.39 Life Safety Point Object

3.39.1 Base Requirements

Base requirements must be met by any IUT that can contain Life Safety Point objects

BTL-7.3.2.15.X6 - Supports writable Mode Property		
	Test Conditionality	Must be executed.
	Test Directives	The test shall be executed using a Life Safety Point object.
	Testing Hints	
BTL-7.3.2.15.X5 - Support writable Tracking_Value		
	Test Conditionality	If Out_Of_Service can be made TRUE, this test must be executed.
	Test Directives	The test shall be executed using a Life Safety Point object.
	Testing Hints	
BTL-7.3.2.15.X9 - Silenced Property test		
	Test Conditionality	Must be executed.
	Test Directives	The test shall be executed using a Life Safety Point object.
	Testing Hints	
BTL-7.3.2.15.X7 - Support Operation_Expected property		
	Test Conditionality	If IUT is capable of generating event notifications then, it Must be executed.
	Test Directives	The test shall be executed using a Life Safety Point object.
	Testing Hints	

3.39.2 Supports Writable Out_Of_Service Property

The Out_Of_Service property in Life Safety objects contained in the IUT are writable.

135.1-2013 - 7.3.1.1 - Out_Of_Service, Status_Flags, and Reliability Tests		
	Test Conditionality	If Out_Of_Service can be made TRUE, this test must be executed.
	Test Directives	The test shall be executed using a Life Safety Point object.
	Testing Hints	

3.39.3 Support Writable Member_Of Property

BTL-7.3.2.15.X8 - Support Writable Member_Of Property		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

3.39.4 Contains an Object with Reliability_Evaluation_Inhibit Property

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

BTL - 7.3.1.X8.1 - Reliability_Evaluation_Inhibit Test		
	Test 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	

[In BTL Test Plan, Add new section for Life Safety Zone Object as below]

3.40 Life Safety Zone Object

3.40.1 Base Requirements

Base requirements must be met by any IUT that can contain Life Safety Zone objects

BTL-7.3.2.15.X6 - Supports writable Mode property		
	Test Conditionality	Must be executed.
	Test Directives	The test shall be executed using a Life Safety Zone object.
	Testing Hints	
BTL-7.3.2.15.X5 - Support writable Tracking_Value		
	Test Conditionality	If Out_Of_Service can be made TRUE, this test must be executed.
	Test Directives	The test shall be executed using a Life Safety Zone object.
	Testing Hints	
BTL-7.3.2.15.X9 - Silenced Property Test		
	Test Conditionality	Must be executed.
	Test Directives	The test shall be executed using a Life Safety Zone object.
	Testing Hints	
BTL-7.3.2.15.X7 - Support Operation_Expected property		
	Test Conditionality	If IUT is capable of generating event notifications then, it Must be executed.
	Test Directives	The test shall be executed using a Life Safety Zone object.
	Testing Hints	

3.40.2 Supports Writable Out_Of_Service Property

The Out_Of_Service property in Life Safety objects contained in the IUT are writable.

135.1-2013 - 7.3.1.1 - Out_Of_Service, Status_Flags, and Reliability Tests		
	Test Conditionality	If Out_Of_Service can be made TRUE, this test must be executed.
	Test Directives	The test shall be executed using a Life Safety Zone object.
	Testing Hints	

3.40.3 Support Writable Member_Of Property

BTL-7.3.2.15.X8 - Support Writable Member_Of property		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

3.40.4 Contains an Object with Reliability_Evaluation_Inhibit Property

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

BTL - 7.3.1.X8.1 - Reliability_Evaluation_Inhibit Test		
	Test 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	

[In BTL Test Plan, Add section to Alarm and Event - Notification - A]

5.1.X Processes CHANGE_OF_LIFE_SAFETY Notifications

The IUT is capable of executing event notifications that convey a CHANGE_OF_LIFE_SAFETY event transition.

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	Must be executed.
	Test Directives	
	Testing Hints	At least one of the tests must be executed with the Event Type set to CHANGE_OF_LIFE_SAFETY.

[In BTL Test Plan, Add section to Alarm and Event - Life Safety - A]

5.21 Alarm and Event - Life Safety - A

5.21.1 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	

5.21.2 Initiates LifeSafetyOperation Requests

135.1-2013 - 8.9.1 - LifeSafetyOperation Service Initiation Tests to an Object		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
135.1-2013 - 8.9.2 - LifeSafetyOperation Service Initiation Tests to all Objects in a Device		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

5.21.3 Executes ConfirmedEventNotifications

The IUT is capable of executing ConfirmedEventNotifications. 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	

5.21.4 Executes UnconfirmedEventNotifications

The IUT is capable of executing UnconfirmedEventNotifications. 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	

5.21.5 Processes Intrinsically Generated Notifications

The IUT is capable of executing ConfirmedEventNotifications 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	
	Testing Hints	

5.21.6 Processes Algorithmically Generated Notifications

The IUT is capable of executing ConfirmedEventNotifications 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	
	Testing Hints	

5.21.7 Processes Event Notifications with Timestamps of the BACnetDateTime Form

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

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

5.21.8 Processes Event Notifications with Timestamps of the Time Form

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

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

5.21.9 Processes Event Notifications with Timestamps of the Sequence Number Form

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

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

5.21.10 Supports AE-ACK-A

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

5.21.11 Supports AE-INFO-A

The IUT must support AE-INFO-A if it claims support for AE-LS-A.

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

5.21.12 Supports AE-AS-A

The IUT must support AE-AS-A if it claims support for AE-LS-A.

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

[In BTL Test Plan, Add section to Alarm and Event - Life Safety - B]

5.22 Alarm and Event - Life Safety - B

5.22.1 Base Requirements

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

BTL - 7.3.1.10.1 - Event_Enable Tests for TO_OFFNORMAL and TO_NORMAL		
	Test Conditionality	If the IUT cannot be configured to meet the configuration requirements then this test shall be skipped.
	Test Directives	If Event Enrollment objects are supported, ensure this functionality is tested on Event Enrollment objects.
	Testing Hints	The BTL will apply this to a single object. The pretester should apply it to all objects that support alarm generation.
135.1-2013 - 7.3.1.12 - Notify_Type Test		
	Test Conditionality	If the IUT cannot be configured to meet the 135.1-2013 configuration requirements then this test shall be skipped.
	Test Directives	If Event Enrollment objects are supported, ensure this functionality is tested on Event Enrollment objects.
	Testing Hints	
135.1-2013 - 8.4 - ConfirmedEventNotification Service Initiation Tests		
	Test Conditionality	Must be executed. Any of the 8.4 tests can be used to ensure that the IUT properly generates ConfirmedEventNotification requests. The specific tests that can be executed are detailed under the test cases for the specific

		algorithms. As long as one of the tests is executed using ConfirmedEventNotifications, then this test case shall be satisfied.
	Test Directives	
	Testing Hints	
135.1-2013 - 8.5 - UnconfirmedEventNotification Service Initiation Tests		
	Test Conditionality	Must be executed. Any of the 8.5 tests can be used to ensure that the IUT properly generates UnconfirmedEventNotification requests. The specific tests that can be executed are detailed under the test cases for the specific algorithms. As long as one of the tests is executed using UnconfirmedEventNotifications, then this test case shall be satisfied.
	Test Directives	
	Testing Hints	
BTL - 7.3.1.X9.1 - Event_Detection_Enable Inhibits Event Generation		
	Test Conditionality	If Protocol_Revision < 13, then this test shall be skipped.
	Test Directives	The objects selected by the tester should include all variants that differ in the set of supported alarming properties, or the writability of any of those properties. At least one instance of each variant should be selected.
	Testing Hints	
BTL - 7.3.1.X9.2 - Event_Detection_Enable Inhibits FAULT		
	Test Conditionality	If Protocol_Revision < 13, then this test shall be skipped.
	Test Directives	The objects selected by the tester should include all variants that differ in the set of supported alarming properties, or the writability of any of those properties. At least one instance of each variant should be selected.
	Testing Hints	
BTL - 7.3.1.X6.1 - Event_Algorithm_Inhibit Test		
	Test Conditionality	If the IUT has no object in which the Event_Algorithm_Inhibit property is present and does not support the Event_Algorithm_Inhibit_Ref property, or has no object in which Event_Detection_Enable can be made TRUE, this test shall be skipped. If the IUT cannot be configured to contain any object capable of an event transition, then this test shall be skipped.
	Test Directives	The object types selected by the tester should include all variants that differ in the set of supported alarming properties, or the writability of any of those properties. At least one instance of each variant should be selected.
	Testing Hints	
BTL - 7.3.1.X7.1 - Event_Algorithm_Inhibit_Ref Test		
	Test Conditionality	If the IUT has no object in which the Event_Algorithm_Inhibit_Ref property is present or has no object in which Event_Detection_Enable can be made TRUE, this test shall be skipped.
	Test Directives	The object types selected by the tester should include all variants that differ in the set of supported alarming properties, or the writability of any of those properties. At least one instance of each variant should be selected.
	Testing Hints	
BTL - 7.3.1.X7.2 - Event_Algorithm_Inhibit Writable Test		
	Test Conditionality	If the IUT has no object in which the Event_Algorithm_Inhibit_Ref property is absent or can be made uninitialized or has no object in which Event_Detection_Enable can be made TRUE, this test shall be skipped.
	Test Directives	The object types selected by the tester should include all variants that differ in the set of supported alarming properties, or the writability of any of those properties. At least one instance of each variant should be selected.
	Testing Hints	
135.1-2013 - 9.9.1 - Reset Single Object Execution Tests		
	Test Conditionality	Must be executed.

	Test Directives	
	Testing Hints	
135.1-2013 - 9.9.2 - Reset Multiple Object Execution Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
135.1-2013 - 9.9.3 - Silencing Execution Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

5.22.2 Supports the Notification Class Object

The IUT supports the Notification Class object in order to send notifications.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for the Notification Class Object in the Checklist.
	Testing Hints	

5.22.3 Supports AE-INFO-B

The IUT must support AE-INFO-B if it claims support for AE-LS-B.

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

5.22.4 Implements Intrinsic Alarming

The IUT contains, or can be made to contain, an object other than an Event Enrollment object that can generate ConfirmedEventNotifications and UnconfirmedEventNotifications.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	This functionality will be tested by the clause 8.4 or 8.5 algorithm tests listed later in this section.
	Testing Hints	

5.22.5 Supports the Event Enrollment Object

The IUT contains, or can be made to contain an Event Enrollment object that can generate ConfirmedEventNotifications and UnconfirmedEventNotifications.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Ensure this functionality is tested on Event Enrollment objects bythe clause 8.4 or 8.5 algorithm tests listed later in this section.
	Testing Hints	

5.22.6Supports the CHANGE_OF_LIFE_SAFETY Algorithm in Event_Parameters

The IUT contains, or can be made to contain an Event Enrollment object that can generate CHANGE_OF_LIFE_SAFETY ConfirmedEventNotifications and UnconfirmedEventNotifications.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Ensure this functionality is tested on Event Enrollment objects by the clause 8.4.8 or 8.5.8 tests in that section.
	Testing Hints	

5.22.7 Supports AE-ACK-B

The IUT initiates EventNotifications with service parameter AckRequired = True.

135.1-2013 - 9.8.5 - Event Information Based on Acknowledged_Transitions		
	Test Conditionality	If the IUT cannot be configured to contain any object with an unacknowledged event, then this test shall be skipped.
	Test Directives	
	Testing Hints	
Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	If the IUT cannot be configured to contain any object with an unacknowledged event, then this test shall be skipped.
	Testing Hints	

5.22.8 Generates Event Notifications with Timestamps of the BACnetDateTime Form

The IUT generates, or can be made to generate, ConfirmedEventNotifications with the Time Stamp parameter taking the BACnetDateTime form.

135.1-2013 - 8.4 - ConfirmedEventNotification Service Initiation Tests		
	Test Conditionality	If the IUT supports AE-N-I-B, these tests may be skipped. Any of the 8.4 tests can be used to ensure that the IUT properly generates ConfirmedEventNotification requests using the BACnetDateTime form. The specific tests that can be executed are detailed under the test cases for the specific algorithms. As long as one of the tests is executed using ConfirmedEventNotifications and the notification that is generated contains a timestamp of the BACnetDateTime form, then this test case shall be satisfied.
	Test Directives	
	Testing Hints	

5.22.9 Generates Event Notifications with Timestamps of the Time Form

The IUT generates, or can be made to generate, ConfirmedEventNotifications with the Time Stamp parameter taking the Time form.

135.1-2013 - 8.4 - ConfirmedEventNotification Service Initiation Tests		
	Test Conditionality	If the IUT supports AE-N-I-B, these tests may be skipped. Any of the 8.4 tests can be used to ensure that the IUT properly generates ConfirmedEventNotification requests using the Time form. The specific tests that can be executed are detailed under the test cases for the specific algorithms. As long as one of the tests is executed using ConfirmedEventNotifications and the notification that is generated contains a timestamp of the Time form, then this test case shall be satisfied.
	Test Directives	
	Testing Hints	

5.22.10 Generates Event Notifications with Timestamps of the Sequence Number Form

The IUT generates, or can be made to generate, ConfirmedEventNotifications with the Time Stamp parameter taking the Sequence Number form.

135.1-2013 - 8.4 - ConfirmedEventNotification Service Initiation Tests		
	Test Conditionality	If the IUT supports AE-N-I-B, these tests may be skipped. Any of the 8.4 tests can be used to ensure that the IUT properly generates ConfirmedEventNotification requests using the Sequence Number form. The specific tests that can be executed are detailed under the test cases for the specific algorithms. As long as one of the tests is

		executed using ConfirmedEventNotifications and the notification that is generated contains a timestamp of the Sequence Number form, then this test case shall be satisfied.
	Test Directives	
	Testing Hints	

5.22.11 Mode Transition Tests when Event State is Maintained

135.1-2013 - 8.4.8.7 Mode Transition Tests when Event State is Maintained		
	Test Conditionality	Must be executed
	Test Directives	
	Testing Hints	

5.22.12 Supports Event_Message_Texts Property

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

BTL - 7.3.1.X4 - Event_Message_Texts Tests		
	Test Conditionality	Must be executed.
	Test Directives	Repeat test once for each object type in the IUT that contains an Event_Message_Texts property.
	Testing Hints	

5.22.13 Supports Event_Message_Texts_Config Property

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

BTL - 7.3.1.X5 - Event_Message_Texts_Config Test		
	Test Conditionality	Must be executed.
	Test Directives	Repeat for each supported transition type (TO_OFFNORMAL, TO_FAULT, TO_NORMAL). Different objects may be selected for different transitions.
	Testing Hints	

[In BTL Specified Test, Add the following new tests]

7.3.2.15.X5 Writable Tracking_Value
BACnet Reference Clauses: 12.15.4, 12.15.5, 12.16.4 and 12.16.5

Purpose: This test case verifies that Tracking_Value is writable when Out_Of_Service is TRUE.

Test Concept: It verifies the interrelationship between the Tracking_Value, Status Flags and Present_Value properties. If the Out_Of_Service property of the object under test is not writable, and the value of the property cannot be changed by other means, then this test shall be omitted. This test applies to Life Safety Zone and Life Safety point object.

The IUT will select one instance of each appropriate object type and test it as described.

Test Steps:

1. MAKE (Out_Of_Service TRUE)
2. VERIFY Out_Of_Service = TRUE
3. VERIFY Status_Flags = (?, FALSE, ?, TRUE)
4. MAKE (Event_State = Normal)
5. VERIFY Event_State = Normal
6. WRITE Tracking_Value = X (any value that corresponds to an Event_State of NORMAL)
7. VERIFY Tracking_Value = X
8. VERIFY Present_Value = X

7.3.2.15.X6 Supports Writable Mode property

BACnet Reference Clauses: 12.15.12, 12.15.13, 12.16.12 and 12.16.13

Purpose: To verify that the Mode property takes one of the values found in the Accepted_Modes property.

Test Concept: It verifies the interrelationship between the Mode, and Accepted_Modes properties. This test applies to Life Safety Zone and Life Safety point object. The IUT will select one instance of each appropriate object type and test it as described.

Test Steps:

```
1.  READ AM = Accepted_Modes
2.  TRANSMIT WriteProperty-Request
        'Object Identifier' =       (the object being tested),
        'Property Identifier' =     Mode,
        'Property Value' =          X (Any valid value from list of AM)
3.  RECEIVE SimpleACK-PDU
4.  VERIFY Mode = X
5.  TRANSMIT WriteProperty-Request
        'Object Identifier' =       (the object being tested),
        'Property Identifier'=      Mode,
        'Property Value' =          X (Any invalid value, which is not present in AM)
6.  RECEIVE BACnet-Error-PDU,
        'Error Class' =     PROPERTY,
        'Error Code' =      VALUE_OUT_OF_RANGE
```

7.3.2.15.X7 Support Operation_Expected Property
BACnet Reference Clauses: 12.15.24, and 12.16.24

Purpose: To verify that the Operation_Expected property takes on the value of ConfirmedEventNotification-Request.

Test Concept: It verifies the interrelationship between the Operation_Expected property, and ConfirmedEventNotification-Request. This test applies to Life Safety Zone and Life Safety point object. The IUT will select one instance of each appropriate object type and test it as described.

Test Steps:

```
1.  MAKE (the IUT send an ConfirmedEventNotification)
2.  RECEIVE ConfirmedEventNotification-Request,
        'Process Identifier' =          (any valid process identifier),
        'Initiating Device Identifier' =   TD,
        'Event Object Identifier' =        (any Life-Safety object),
        'Time Stamp' =                     (the current local time),
        'Notification Class' =             (any valid notification class),
        'Priority' =                       (any valid priority),
        'Event Type' =                     CHANGE-OF-LIFE-SAFETY,
        'Message Text' =                   (any character string),
        'Notify Type' =                    ALARM | EVENT,
        'AckRequired' =                    TRUE |FALSE,
        'From State' =                     NORMAL,
        'To State' =                       (any non-normal state appropriate to the event type),
        'Event Values' =                   (New State: (Any Valid State), New-Mode: (Any Valid Mode),
                                               Status-Flag: (TRUE, FALSE, ?, ?),
                                               Operation_Expected: (X: Any Valid operation))
3.  VERIFY Operation_Expected = X ( operation expected in the step 2)
```

7.3.2.15.X8 Support Writable Member_Of property

BACnet Reference Clauses: 12.15.29, and 12.16.29

Purpose: To verify that the Member_Of property takes only supported values of the life safety objects within the IUT.

Test Concept: If the property is writable and is restricted to referencing objects within the containing device, an attempt to write a reference to an object outside the containing device into this property shall cause a Result (-), if the property is not writable and if the value of the property cannot be changed by other means, then this test shall be omitted. The IUT will select one instance of each appropriate object type and test it as described.

Test Steps:

```
1.  TRANSMIT WriteProperty-Request,
        'Object Identifier' =    (life safety object),
        'Property Identifier' =  Member_Of
        'Property Value' =       X (any valid life safety object)
2.  RECEIVE Simple-ACK-PDU,
3.  TRANSMIT ReadProperty-Request,
        'Object Identifier' =    (life safety object),
        'Property Identifier' =  Member_Of
4.  RECEIVE ReadProperty-ACK,
        'Object Identifier' =    (the object being tested),
        'Property Identifier' =  Member_Of
        'Property Value' =       X (the value used in step 1)
```

7.3.2.15.X9 Silenced Property test

Purpose: This test verifies the behavior of Silenced property.

Test Concept: Verify the interrelationship between the Silenced property and any audible or visual indication that has been silenced by the receipt of a LifeSafetyOperation service request or a local process. If the Silenced property of the object under test is unchanging by means of a LifeSafetyOperation service requests, because none of the silencing operations are supported, then this test shall be omitted. This test applies to Life Safety Zone and Life Safety Point object.

The IUT will select one instance of each appropriate object type and test it as described.

Test Steps:

```
1.  InitialSilencedState = READ Silenced
2.  TRANSMIT LifeSafetyOperation-Request,
        'Requesting Process Identifier' =    (any valid identifier),
        'Requesting Source' = (any valid character string),
        'Request' =    (any supported LifeSafetyOperation request transmitted to silence the sounder/strobe),
        'Object Identifier' =    (the selected object)
3.  RECEIVE BACnet-SimpleACK-PDU
4.  CHECK (Sounder/Strobe inactive)
5.  ResultingSilencedState = READ Silenced
6.  CHECK (the ResultingSilencedState is equal to the InitialSilencedState, modified by the LifeSafetyOperation request
transmitted)
```

BTL-15.2h-4: Channel Object Tests [BTLWG-32]

Overview:

The tests for Channel object added to the standard in Protocol_Revision 14 need tests.

Changes:

[In BTL Checklist, Add new Channel Object section]

Support	Listing	Option
Channel Object		
	R	Base Requirements
	S	Supports writable Out_Of_Service properties
	O	Supports an object with Reliability_Evaluation_Inhibit Property
	O	Supports Allow_Group_Delay_Inhibit
	R	Is able to correctly handle targets of BOOLEAN data type
	R	Is able to correctly handle targets of Unsigned data type
	R	Is able to correctly handle targets of INTEGER data type
	R	Is able to correctly handle targets of REAL data type
	R	Is able to correctly handle targets of Double data type
	R	Is able to correctly handle targets of ENUMERATED data type
	R	Is able to correctly handle targets of NULL data type
	R	Is able to correctly handle targets of OCTET STRING data type
	R	Is able to correctly handle targets of CharacterString data type
	R	Is able to correctly handle targets of BIT STRING data type
	R	Is able to correctly handle targets of Date data type
	R	Is able to correctly handle targets of Time data type
	R	Is able to correctly handle targets of BACnetObjectIdentifier data type
	R	Is able to correctly handle targets of BACnetLightingCommand data type
	R	Supports DS-WG-I-B
	O	Supports DS-WG-E-B

[In Test Plan sections 3.X add new sections, as shown]

3.53 Channel Objects

3.53.1 Base Requirements

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

BTL - 7.3.2.X40.2 - Last_Priority Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 7.3.2.X40.3 - WriteGroup Service Support Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 7.3.2.X40.4 - Propagation Entirety Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 7.3.2.X40.5 - Write_Status Test		

	Test Conditionality	Must be executed.
	Test Directives	Repeat the test by sending a write group request to both BROADCAST and to the IUT.
	Testing Hints	
BTL - 7.3.2.X40.12 - Write Priority Test		
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test by writing to the Present_Value of the Channel object using WritePropertyMultiple and WriteGroup service if supported by the IUT.
	Testing Hints	
BTL - 7.3.2.X40.13 - Writing with a NULL value Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

3.53.2 Supports Writable Out_Of_Service Property

The Out_Of_Service property in Channel objects contained in the IUT are writable.

135.1-2013 - 7.3.1.1 - Out_Of_Service, Status_Flags, and Reliability Tests		
	Test Conditionality	If this property is writable, this test must be executed.
	Test Directives	
	Testing Hints	

3.53.3 Supports an Object with Reliability_Evaluation_Inhibit Property

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

BTL - 7.3.1.X8.1 - Reliability_Evaluation_Inhibit Test		
	Test 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	

3.53.4 Supports Allow_Group_Delay_Inhibit

The object contains Allow_Group_Delay_Inhibit property.

BTL - 7.3.2.X40.6 - Allow_Group_Delay_Inhibit Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

3.53.5 Is Able to Correctly Handle Targets of BOOLEAN Data Type

The channel object on the IUT is able to correctly handle targets of BOOLEAN data type.

BTL - 7.3.2.X40.7 - Numeric to BOOLEAN Coercion Rule Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. If the IUT does not contain or cannot be made to contain any writable BOOLEAN properties, skip the test with local target.

	Test Directives	Repeat the test by writing all numeric data type (Unsigned, INTEGER, REAL, Double and ENUMERATED) to a Present_Value of a Channel object containing a BOOLEAN target object property reference.
	Testing Hints	
BTL - 7.3.2.X40.10 - Invalid Datatype Coercion Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. Skip the test for (an) invalid data type(s) not supported by the device containing a target object property.
	Test Directives	Repeat the test by writing all invalid data type (OCTET STRING, CharacterString, BIT STRING,Date, Time, BACnetObjectIdentifier and BACnetLightingCommand) to a Present_Value of a Channel object containing a BOOLEAN target object property reference.
	Testing Hints	
BTL - 7.3.2.X40.11 - No Coercion Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. If the IUT does not contain or cannot be made to contain any writable object properties of the written value type, skip the test with local target. Select a value within an acceptable range of the target object property.
	Test Directives	Repeat the test by writing all data type that does not require coercion (NULL, BOOLEAN) to a Present_Value of a Channel object containing a BOOLEAN target object property reference.
	Testing Hints	

3.53.6 Is Able to Correctly Handle Targets of Unsigned Data Type

The channel object on the IUT is able to correctly handle targets of Unsigned data type.

BTL - 7.3.2.X40.8 - BOOLEAN to Numeric Coercion Rule Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. If the IUT does not contain or cannot be made to contain any writable Numeric properties, skip the test with local target.
	Test Directives	Test with Unsigned target object reference.
	Testing Hints	
BTL - 7.3.2.X40.9 - Unsigned/INTEGER/REAL/Double to Numeric Coercion Rule Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. If the IUT does not contain or cannot be made to contain any writable Numeric properties, skip the test with local target. Select a value within an acceptable range of the target object property.
	Test Directives	Repeat the test by writing all numeric data type (Unsigned, INTEGER, REAL, Double and ENUMERATED) to a Present_Value of a Channel object containing an Unsigned target object property reference. When writing either unsigned or INTEGER data type, use a value less than 2147483647 as par Coercion Rule 3-6 defined in ASHRAE 135. When writing either REAL or Double data type, use a value less than 2147483000 as par Coercion Rule 3-6 defined in ASHRAE 135.
	Testing Hints	
BTL - 7.3.2.X40.10 - Invalid Datatype Coercion Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. If the IUT does not contain or cannot be made to contain any writable Numeric properties, skip the test with local target
	Test Directives	Repeat the test by writing all invalid data type (OCTET STRING, CharacterString, BIT STRING,Date, Time and BACnetLightingCommand) to a Present_Value of a Channel object containing an Unsigned target object property reference.

	Testing Hints	
BTL - 7.3.2.X40.11 - No Coercion Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. If the IUT does not contain or cannot be made to contain any writable object properties of the written value type, skip the test with local target. Select a value within an acceptable range of the target object property.
	Test Directives	Repeat the test by writing all data type that does not require coercion (NULL,Unsigned, ENUMERATED, and BACnetObjectIdentifier) to a Present_Value of a Channel object containing an Unsigned target object property reference.
	Testing Hints	

3.53.6 Is Able to Correctly Handle Targets of INTEGER Data Type

The channel object on the IUT is able to correctly handle targets of INTEGER data type.

BTL - 7.3.2.X40.8 - BOOLEAN to Numeric Coercion Rule Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. If the IUT does not contain or cannot be made to contain any writable INTEGER properties, skip the test with local target.
	Test Directives	Test with INTEGER target object reference.
	Testing Hints	
BTL - 7.3.2.X40.9 - Unsigned/INTEGER/REAL/Double to Numeric Coercion Rule Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. If the IUT does not contain or cannot be made to contain any writable Numeric properties, skip the test with local target. Select a value within an acceptable range of the target object property.
	Test Directives	Repeat the test by writing all numeric data type (Unsigned, INTEGER, REAL, Double and ENUMERATED) to a Present_Value of a Channel object containing an INTEGER target object property reference. When writing either REAL or DOUBLE data type, use a value more than -2147483000 and less than 214783000 as par Coercion Rule 3-6 defined in ASHRAE 135.
	Testing Hints	
BTL - 7.3.2.X40.10 - Invalid Datatype Coercion Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. Skip the test for (an) invalid data type(s) not supported by the device containing a target object property.
	Test Directives	Repeat the test by writing all invalid data type (OCTET STRING, CharacterString, BIT STRING,Date, Time, BACnetObjectIdentifier and BACnetLightingCommand) to a Present_Value of a Channel object containing an INTEGER target object property reference.
	Testing Hints	
BTL - 7.3.2.X40.11 - No Coercion Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. If the IUT does not contain or cannot be made to contain any writable object properties of the written value type, skip the test with local target. Select a value within an acceptable range of the target object property.
	Test Directives	Repeat the test by writing all data type that does not require coercion (NULL and INTEGER) to a Present_Value of a Channel object containing an INTEGER target object property reference.
	Testing Hints	

3.53.6 Is Able to Correctly Handle Targets of REAL Data Type

The channel object on the IUT is able to correctly handle targets of REAL data type.

BTL - 7.3.2.X40.8 - BOOLEAN to Numeric Coercion Rule Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. If the IUT does not contain or cannot be made to contain any writable BOOLEAN properties, skip the test with local target.
	Test Directives	Test with REAL target object property reference.
	Testing Hints	
BTL - 7.3.2.X40.9 - Unsigned/INTEGER/REAL/Double to Numeric Coercion Rule Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. If the IUT does not contain or cannot be made to contain any writable Numeric properties, skip the test with local target. Select a value within an acceptable range of the target object property.
	Test Directives	Repeat the test by writing all numeric datatype (Unsigned, INTEGER, REAL, Double and ENUMERATED) to a Present_Value of a Channel object containing a REAL target object property reference. When writing either Unsigned or INTEGER data type, use a value in precision up to seven significant digits as par Coercion Rules defined in ASHRAE 135. When writing DOUBLE data type, use a value less than 3.4*10+38 and more than 3.4*10-38 as par Coercion Rules defined in ASHRAE 135.
	Testing Hints	
BTL - 7.3.2.X40.10 - Invalid Datatype Coercion Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. Skip the test for (an) invalid data type(s) not supported by the device containing a target object property.
	Test Directives	Repeat the test by writing all invalid data type (OCTET STRING, CharacterString, BIT STRING,Date, Time, BACnetObjectIdentifier and BACnetLightingCommand) to a Present_Value of a Channel object containing an REAL target object property reference.
	Testing Hints	
BTL - 7.3.2.X40.11 - No Coercion Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. If the IUT does not contain or cannot be made to contain any writable object properties of the written value type, skip the test with local target. Select a value within an acceptable range of the target object property.
	Test Directives	Repeat the test by writing all data type that does not require coercion (NULL and REAL) to a Present_Value of a Channel object containing a REAL target object property reference.
	Testing Hints	

3.53.6 Is Able to Correctly Handle Targets of Double Data Type

The channel object on the IUT is able to correctly handle targets of DOUBLE data type.

BTL - 7.3.2.X40.8 - BOOLEAN to Numeric Coercion Rule Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. If the IUT does not contain or cannot be made to contain any writable BOOLEAN properties, skip the test with local target.

	Test Directives	Test with Double target object property reference.
	Testing Hints	
BTL - 7.3.2.X40.9 - Unsigned/INTEGER/REAL/Double to Numeric Coercion Rule Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. If the IUT does not contain or cannot be made to contain any writable Numeric properties, skip the test with local target. Select a value within an acceptable range of the target object property.
	Test Directives	Repeat the test by writing all numeric datatype (Unsigned, INTEGER, REAL, Double and ENUMERATED) to a Present_Value of a Channel object containing a Double target object property reference.
	Testing Hints	
BTL - 7.3.2.X40.10 - Invalid Datatype Coercion Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. Skip the test for (an) invalid data type(s) not supported by the device containing a target object property.
	Test Directives	Repeat the test by writing all invalid data type (OCTET STRING, CharacterString, BIT STRING,Date, Time, BACnetObjectIdentifier and BACnetLightingCommand) to a Present_Value of a Channel object containing a Double target object property reference.
	Testing Hints	
BTL - 7.3.2.X40.11 - No Coercion Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. If the IUT does not contain or cannot be made to contain any writable object properties of the written value type, skip the test with local target. Select a value within an acceptable range of the target object property.
	Test Directives	Repeat the test by writing all data type that does not require coercion (NULL and Double) to a Present_Value of a Channel object containing a Double target object property reference.
	Testing Hints	

3.53.6 Is Able to Correctly Handle Targets of Enumerated Data Type

The channel object on the IUT is able to correctly handle targets of Enumerated data type.

BTL - 7.3.2.X40.8 - BOOLEAN to Numeric Coercion Rule Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. If the IUT does not contain or cannot be made to contain any writable Numeric properties, skip the test with local target. Select a value within an acceptable range of the target object property.
	Test Directives	Test with Enumerated target object property reference.
	Testing Hints	
BTL - 7.3.2.X40.9 - Unsigned/INTEGER/REAL/Double to Numeric Coercion Rule Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. If the IUT does not contain or cannot be made to contain any writable Numeric properties, skip the test with local target. Select a value within an acceptable range of the target object property.
	Test Directives	Repeat the test by writing all numeric datatype (Unsigned, INTEGER, REAL, Double and Enumerated) to a Present_Value of a Channel object containing an Enumerated target object property reference.
	Testing Hints	
BTL - 7.3.2.X40.10 - Invalid Datatype Coercion Test		

	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. Skip the test for (an) invalid data type(s) not supported by the device containing a target object property.
	Test Directives	Repeat the test by writing all invalid data type (OCTET STRING, CharacterString, BitString, Date, Time, BACnetObjectIdentifier and BACnetLightingCommand) to a Present_Value of a Channel object containing an Enumerated target object property reference.
	Testing Hints	
BTL - 7.3.2.X40.11 - No Coercion Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. If the IUT does not contain or cannot be made to contain any writable object properties of the written value type, skip the test with local target. Select a value within an acceptable range of the target object property.
	Test Directives	Repeat the test by writing all data type that does not require coercion (NULL, Unsigned and Enumerated) to a Present_Value of a Channel object containing an Enumerated target object property reference.
	Testing Hints	

3.53.7 Is able to correctly handle targets of OCTET STRING data type

The channel object on the IUT is able to correctly handle targets of OCTET STRING data type

BTL - 7.3.2.X40.10 - Invalid Datatype Coercion Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. Skip the test for (an) invalid data type(s) not supported by the device containing a target object property.
	Test Directives	Repeat the test by writing all invalid data type (BOOLEAN, Unsigned, INTEGER, REAL, Double, CharacterString, BitString, Enumerated, Date, Time, BACnetObjectIdentifier and BACnetLightingCommand) to a Present_Value of a Channel object containing an OCTET STRING target object property reference.
	Testing Hints	
BTL - 7.3.2.X40.11 - No Coercion Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. If the IUT does not contain or cannot be made to contain any writable object properties of the written value type, skip the test with local target. Select a value within an acceptable range of the target object property.
	Test Directives	Repeat the test by writing all data type that does not require coercion (NULL and OCTET STRING) to a Present_Value of a Channel object containing an OCTET STRING target object property reference.
	Testing Hints	

3.53.8 Is able to correctly handle targets of Character String data type

The channel object on the IUT is able to correctly handle targets of Character String data type.

BTL - 7.3.2.X40.10 - Invalid Datatype Coercion Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. Skip the test for (an) invalid data type(s) not supported by the device containing a target object property.

	Test Directives	Repeat the test by writing all invalid data type (BOOLEAN, Unsigned, INTEGER, REAL, Double, OCTET STRING, BitString, Enumerated, Date, Time, BACnetObjectIdentifier and BACnetLightingCommand) to a Present_Value of a Channel object containing a Character String target object property reference.
	Testing Hints	
BTL - 7.3.2.X40.11 - No Coercion Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. If the IUT does not contain or cannot be made to contain any writable object properties of the written value type, skip the test with local target. Select a value within an acceptable range of the target object property.
	Test Directives	Repeat the test by writing all data type that does not require coercion (NULL and Character String) to a Present_Value of a Channel object containing a Character String target object property reference.
	Testing Hints	

3.53.9 Is Able to Correctly Handle Targets of BitString Data Type

The channel object on the IUT is able to correctly handle targets of BITSTRING data type.

BTL - 7.3.2.X40.10 - Invalid Datatype Coercion Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. Skip the test for (an) invalid data type(s) not supported by the device containing a target object property.
	Test Directives	Repeat the test by writing all invalid data type (BOOLEAN, Unsigned, INTEGER, REAL, Double, OCTET STRING, Character String, Enumerated, Date, Time, BACnetObjectIdentifier and BACnetLightingCommand) to a Present_Value of a Channel object containing a BIT STRING target object property reference.
	Testing Hints	
BTL - 7.3.2.X40.11 - No Coercion Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. If the IUT does not contain or cannot be made to contain any writable object properties of the written value type, skip the test with local target. Select a value within an acceptable range of the target object property.
	Test Directives	Repeat the test by writing all data type that does not require coercion (NULL and BitString) to a Present_Value of a Channel object containing a BitString target object property reference.
	Testing Hints	

3.53.11 Is Able to Correctly Handle Targets of Date Data Type

The channel object on the IUT is able to correctly handle targets of Date data type

BTL - 7.3.2.X40.10 - Invalid Datatype Coercion Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. Skip the test for (an) invalid data type(s) not supported by the device containing a target object property.
	Test Directives	Repeat the test by writing all invalid data type (BOOLEAN, Unsigned, INTEGER, REAL, Double, OCTET STRING, Character String, BitString, Enumerated, Time, BACnetObjectIdentifier and BACnetLightingCommand) to a Present_Value of a Channel object containing a Date target object property reference.
	Testing Hints	
BTL - 7.3.2.X40.11 - No Coercion Test		

	Test Conditionality	
	Test Directives	Repeat the test by writing all data type that does not require coercion (NULL and Date) to a Present_Value of a Channel object containing a Date target object property reference.
	Testing Hints	

3.53.12 Is Able to Correctly Handle Targets of Time Data Type

The channel object on the IUT is able to correctly handle targets of Time data type.

BTL - 7.3.2.X4010 - Invalid Datatype Coercion Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. Skip the test for (an) invalid data type(s) not supported by the device containing a target object property.
	Test Directives	Repeat the test by writing all invalid data type (BOOLEAN, Unsigned, INTEGER, REAL, Double, OCTET STRING, Character String, BitString, Enumerated, Date, BACnetObjectIdentifier and BACnetLightingCommand) to a Present_Value of a Channel object containing a Time target object property reference.
	Testing Hints	
BTL - 7.3.2.X40.11 - No Coercion Test		
	Test Conditionality	
	Test Directives	Repeat the test by writing all data type that does not require coercion (NULL and Time) to a Present_Value of a Channel object containing an OCTET STRING target object property reference.
	Testing Hints	

3.53.13 Is Able to Correctly Handle Targets of BACnetObjectIdentifier Data Type

The channel object on the IUT is able to correctly handle targets of BACnetObjectIdentifier data type.

BTL - 7.3.2.X40.10 - Invalid Datatype Coercion Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. Skip the test for (an) invalid data type(s) not supported by the device containing a target object property.
	Test Directives	Repeat the test by writing all invalid data type (BOOLEAN, INTEGER, REAL, Double, OCTET STRING, Character String, BitString, Enumerated, Date, Time and BACnetLightingCommand) to a Present_Value of a Channel object containing a BACnetObjectIdentifier target object property reference.
	Testing Hints	
BTL - 7.3.2.X40.11 - No Coercion Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. If the IUT does not contain or cannot be made to contain any writable object properties of the written value type, skip the test with local target. Select a value within an acceptable range of the target object property.
	Test Directives	Repeat the test by writing data type that does not require coercion (NULL, Unsigned and BACnetObjectIdentifier) to a Present_Value of a Channel object containing a BACnetObjectIdentifier target object property reference.
	Testing Hints	

3.53.14 Is Able to Correctly Handle Targets of BACnetLightingCommand Data Type

The channel object on the IUT is able to correctly handle targets of BACnetLighting Command data type.

BTL - 7.3.2.X40.10 - Invalid Datatype Coercion Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. Skip the test for (an) invalid data type(s) not supported by the device containing a target object property.
	Test Directives	Repeat the test by writing all invalid data type (NULL, BOOLEAN, Unsigned, INTEGER, REAL, Double, OCTET STRING, Character String, BitString, Enumerated, Date, Time and BACnetObjectIdentifier) to a Present_Value of a Channel object containing a BACnetLightingCommand target object property reference.
	Testing Hints	
BTL - 7.3.2.X40.11 - No Coercion Test		
	Test Conditionality	Test with both local and remote target. If DS-WG-E-B is not supported, test only with local target. If the IUT does not contain or cannot be made to contain any writable object properties of the written value type, skip the test with local target. Select a value within an acceptable range of the target object property.
	Test Directives	Repeat the test by writing BACnetLightingCommand value to a Present_Value of a Channel object containing a BACnetObjectIdentifier target object property reference.
	Testing Hints	

3.53.15 Supports DS-WG-I-B

The IUT supports DS-WG-I-B in order to execute a WriteGroup service

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-WG-I-B in the checklist
	Testing Hints	

3.53.16 Supports DS-WG-E-B

The IUT supports DS-WG-E-B in order to execute a WriteGroup service and then to initiate a write service to propagate a value to one or more target object on one or more remote device.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-WG-E-B in the checklist
	Testing Hints	

[Add Clause 6.3.X1 and 4.5.9 to the BTL Specified Tests]

6.3.X1 Channel Write Fail Time

The Channel Write Fail Time is the elapsed time, in seconds, between a change to the Present_Value of a Channel object and when a test is considered to have failed because the first write operation associated with the newly written value state has not been performed. If the Channel object has multiple target properties to write to, the time to write all of them would be less than or equal to the number of target properties times this value.

4.5.9 Timers

This section defines timer values that are used to determine when a test has failed because an appropriate response has not been observed by the TD. A Real value in seconds must be provided for each timer. See 6.3.

Fail Times: ↵
{↵
Notification Fail Time: ❑↵
Internal Processing Fail Time: ❑↵
Minimum ON/OFF Time: ❑↵
Schedule Evaluation Fail Time: ❑↵
External Command Fail Time: ❑↵
Program Object State Change Fail Time: ❑↵
Acknowledgement Fail Time: ❑↵
Slave Proxy Confirm Interval: ❑↵
Unconfirmed Response Fail Time: ❑↵
Channel Write Fail Time: ❑↵

}↵

[In BTL Specified Tests, add Channel object specific tests in section 7.3.2]

7.3.2.X40.2 Last_Priority Test

Purpose: To verify that the initial value of Last_Priority is 16. To verify that a Channel object correctly retains the priority of written values. To verify that a Last_Priority will have a default priority of 16 if the last attempt to write to the Present_Value was done without specifying the priority.

Test Concept: First, confirm that the default value of Last_Priority is 16. Next, write a valid value to a Channel object using a valid priority level other than 16 and then check the value of Last_Priority to make sure it shows the specified priority level. Finally, write a valid value to a Channel object again but without specifying priority and then check the value of Last_Priority to make sure that it now shows 16 again.

Configuration Requirements: Configure entry X of the Channel object's List_Of_Object_Property_References to refer to a writable Present_Value property which is either on local device or remote device.

Test Steps:

1. READ LEN = List_Of_Object_Property_References, ARRAY INDEX = 0
2. READ B = List_Of_Object_Property_References, ARRAY INDEX = X
3. VERIFY Last_Priority = 16
4. WRITE Present_Value = (Any valid value), PRIORITY = (Y: Any valid value < 16)
5. WAIT (**Channel Write Fail Time** * LEN)
6. VERIFY Last_Priority = Y
7. WRITE Present_Value = (Any valid value)
8. WAIT (**Channel Write Fail Time** * LEN)
9. VERIFY Last_Priority = 16

7.3.2.X40.3 WriteGroup Service Support Test

Purpose: To verify that the Present_Value of the Channel object can be written by WriteGroup Service

Test Concept: The Channel object, O1, is written to via WriteGroup and it is verified that the Present_Value of the object is updated correctly.

Test Steps:

```
1.  READ X = O1, Present_Value
2.  TRANSMIT                    WriteGroup-Request,
        'Group Number' =        (one of the Control_Group values configured in O1),
        'Write Priority' =      (any valid value),
        'Change List' =         (O1's channel number, no overriding priority, Y: a value different than X)
3.  VERIFY Present_Value = Y
```

7.3.2.X40.4 Propagation Entirety Test

Purpose: To verify that the Channel object keeps propagating the value to each local/remote object property reference in its List_Of_Object_Property_References after propagating the value fails for one or more target.

Test Concept: The Channel object, O1, is configured with at least 1 referenced target which the Channel object won't send the write to due to an invalid datatype coercion error. The Channel object's Present_Value is written, and the writes to remote targets are checked. Once all writes complete, the Write_Status is verified to be FAILED and all targets the Channel object sent the write to are verified to have accepted the written value.

Configuration Requirements: Configure the Channel object's List_Of_Object_Property_References so that at least one of the target references (contained in entry X of List_Of_Object_Property_References) will result in an invalid datatype coercion when the value WrittenValue is written to the Channel object. The rest of the target reference(s) shall be selected such that they will accept the written value as is without coercion. Refer to the Table 12-63 – Datatype Coercion Rules from ASHRAE 135 for the invalid datatypes.

Test Steps:

```
1.  READ LEN = List_Of_Object_Property_References, ARRAY INDEX = 0
2.  READ N1 = List_Of_Object_Property_References, ARRAY INDEX = X
3.  WRITE Present_Value  = WRITTEN_VALUE
4.  WAIT (Channel Write Fail Time * LEN)
5.  REPEAT REF = (each reference in O1.List_Of_Object_Property_References) {
        IF REF is not contained in the IUT
            RECEIVE WriteProperty-Request
                'Object Identifier' =       (Object Identifier of N1),
                'Property Identifier' =     (Property Identifier of N1),
                'Property Value' =          WrittenValue
            IF REF <> N1 THEN
                TRANSMIT BACnet-SimpleACK-PDU
    }
6.  VERIFY Write_Status = FAILED
7.  REPEAT REF = (each reference in O1.List_Of_Object_Property_References) {
        IF REF <> N1 THEN
            VERIFY REF = WRITTEN_VALUE
    }
```

7.3.2.X40.5 Write_Status Test

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

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

Test Steps:

```
1.  READ LEN = List_Of_Object_Property_References, ARRAY INDEX = 0

-- IDLE test

2.  READ L = List_Of_Object_Property_References
3.  VERIFY L = (empty)
```

4. VERIFY Write_Status = IDLE
5. WRITE Present_Value = (any valid value)
6. VERIFY Write_Status = IDLE

-- IN_PROGRESS and FAIL test

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

-- SUCCESSFUL test

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

7.3.2.X40.6 Allow_Group_Delay_Inhibit Test

Purpose: To verify that no Execution_Delay will be applied to any of the writes if Allow_Group_Delay_Inhibit is TRUE.

Test Concept: Setup List_Of_Object_Property_References to contain 2 valid entries PR1, PR2 and provide each with an execution delay (ED1 and ED2). Set Allow_Group_Delay_Inhibit to TRUE so that no delays will occur between writes to referenced properties. Write to the Channel object's Present_Value and verify that no delay occurs between writes to the referenced properties.
Set Allow_Group_Delay_Inhibit to FALSE so that delays will occur between writes to referenced properties. Write to the Channel object's Present_Value and verify that delays occur between writes to the referenced properties.

Configuration Requirements: PR1 and PR2 shall be references to writable properties and shall be the same datatype. ED1 and ED2 shall be values which are large enough that the delay between writes is sufficient for the test. V1 and V2 shall be of the expected datatype for PR1 and PR2 so that no coercion occur, and shall be different values. This test shall be skipped if the Channel object does not support at least 2 entries in the List_Of_Object_Property_References.

Test Steps:
1. READ LEN = List_Of_Object_Property_References, ARRAY INDEX = 0

-- Setup the Channel object

2. WRITE List_Of_Object_Property_References = (PR1, PR2)
3. WRITE Execution_Delay = (ED1, ED2)

-- Test that delays are inhibited

4. WRITE Allow_Group_Delay_Inhibit = TRUE
5. WRITE Present_Value = V1
6. WAIT (**Channel Write Fail Time** * LEN)
7. VERIFY PR1 = V1
8. VERIFY PR2 = V1
9. VERIFY Write_Status = SUCCESSFUL

-- Test that delays are not inhibited

10. WRITE Allow_Group_Delay_Inhibit = FALSE
11. WRITE Present_Value = V2
12. WAIT (**Channel Write Fail Time** * LEN)
13. VERIFY PR1= V1
14. VERIFY PR2 = V1
15. VERIFY Write_Status = IN_PROGRESS
16. WAIT (ED1)
17. VERIFY PR1 = V2
18. VERIFY PR2 = V1
19. VERIFY Write_Status = IN_PROGRESS

20. WAIT (ED2 – ED1)
21. VERIFY PR2 = V2
22. VERIFY Write_Status = SUCCESSFUL

7.3.2.X40.7 Numeric to BOOLEAN Coercion Rule Test

Purpose: To verify that the Channel object correctly propagates all numeric datatype values to a BOOLEAN target based on Coercion Rule 1 – Numeric to BOOLEAN.

Test Concept: Write a value of 0 is to the Present_Value of the Channel object with a BOOLEAN target object property reference, verify that a target object property has a value of FALSE, and that Write_Status of the Channel object shows SUCCESSFUL. When any non-zero numeric value is written to the Present_Value of the same Channel object, verify that a target object property has a value of TRUE, and a Write Status shows SUCCESSFUL.

Configuration Requirements: Configure entry X of the Channel object's List_Of_Object_Property_References to refer to a writable BOOLEAN object property.

Test Steps:
1. READ LEN = List_Of_Object_Property_References, ARRAY INDEX = 0
2. READ B = List_Of_Object_Property_References, ARRAY INDEX = X
3. WRITE Present_Value = 0
4. WAIT (**Channel Write Fail Time** * LEN)
5. VERIFY B = FALSE
6. VERIFY Write_Status = SUCCESSFUL
7. WRITE Present_Value = (Any non-zero numeric value)
8. WAIT (**Channel Write Fail Time** * LEN)
9. VERIFY B = TRUE
10. VERIFY Write_Status = SUCCESSFUL

7.3.2.X40.8 BOOLEAN to Numeric Coercion Rule Test

Purpose: To verify that the Channel object can correctly propagate BOOLEAN values to a numeric target object property reference based on Coercion Rule 2 – BOOLEAN to Numeric defined in ASHRAE 135.

Test Concept: When a value of FALSE is written to the Present_Value of the Channel object with a numeric target object property reference, verify that the target object property has a value of 0, and a Write_Status of the Channel object shows SUCCESSFUL. When a value of TRUE is written to the present value of the same Channel object, verify that a target object property has a value of 1, and a Write_Status shows SUCCESSFUL.

Configuration Requirements: Configure entry X of the Channel object's List_Of_Object_Property_References to refer to a writable numeric object property on the IUT. The referenced property shall not be 0 at the start of the test.

Test Steps:

1. READ LEN = List_Of_Object_Property_References, ARRAY INDEX = 0
2. READ N = List_Of_Object_Property_References, ARRAY INDEX = X
3. VERIFY N <> 0 -- non-zero so that coercion is verified in the following write
4. WRITE Present_Value = FALSE
5. WAIT (**Channel Write Fail Time** * LEN)
6. VERIFY N = 0
7. VERIFY Write_Status = SUCCESSFUL
8. WRITE Present_Value = TRUE
9. WAIT (**Channel Write Fail Time** * LEN)
10. VERIFY N = 1
11. VERIFY Write_Status = SUCCESSFUL

7.3.2.X40.9 Unsigned/INTEGER/REAL/Double to Numeric Coercion Rule Test

Purpose: To verify that the Channel object correctly propagates Unsigned, INTEGER, REAL or Double datatype values to a numeric target object property reference.

Test Concept: Select a Channel object with a numeric target property, N. Select an Unsigned, INTEGER, REAL or Double value, V1, which is in the acceptable range for N, and which coerces to value V2 based on N's datatype. Write V1 to the Present_Value of the Channel object. Verify that the N changes to V2 and that Write_Status of the Channel object is SUCCESSFUL.

Configuration Requirements: Configure entry X of the Channel object's List_Of_Object_Property_References to refer to the selected numeric property N. Configure the Channel object with no execution delays.

Test Steps:

1. READ LEN = List_Of_Object_Property_References, ARRAY INDEX = 0
2. VERIFY List_Of_Object_Property_References = N, ARRAY INDEX = X
3. WRITE Present_Value = V1
4. WAIT (**Channel Write Fail Time** * LEN)
5. VERIFY N = V2
6. VERIFY Write_Status = SUCCESSFUL

7.3.2.X40.10 Invalid Datatype Coercion Test

Purpose: To check that the Channel object does not write to a target object property reference and the Write_Status indicates FAILED when invalid datatype coercion occur.

Test Concept: When an invalid data type value is written to a Present_Value of the Channel object, verify that a target object reference value does not change.

Configuration Requirements: Configure entry X of the Channel object's List_Of_Object_Property_References to refer to a writable object property of a specific data type on the IUT such that the Channel object will fail to propagate InvalidDataTypeValue. Refer to the Table 12-63 - Datatype Coercion Rules from ASHRAE 135 for the invalid datatypes.

Test Steps:

1. READ LEN = List_Of_Object_Property_References, ARRAY INDEX = 0
2. READ N = List_Of_Object_Property_References, ARRAY INDEX = X
3. WRITE Present_Value = (InvalidDataTypeValue: Any invalid data type value)
4. WAIT (**Channel Write Fail Time** * LEN)
5. VERIFY N != InvalidDataTypeValue
6. VERIFY Write_Status = FAILED

7.3.2.X40.11 No Coercion Test

Purpose: To check that the Channel object can successfully write to a target object property reference without any value conversions and Write_Status indicates SUCCESSFUL when no coercion occurs.

Test Concept: When a valid data type value is written to a Present_Value of the Channel object using a value that require no coercion, verify that a written value is directly mapped to a target object reference and a Write_Status of the Channel object shows SUCCESSFUL.

Configuration Requirements: Configure entry X of the Channel object's List_Of_Object_Property_References to refer to a writable object property of a specific data type on the IUT such that no coercion will occur. Refer to the Table 12-63 – Datatype Coercion Rules from ASHRAE 135 for the data types that require no coercion.

Test Steps:

1. READ LEN = List_Of_Object_Property_References, ARRAY INDEX = 0
2. READ N = List_Of_Object_Property_References, ARRAY INDEX = X
3. WRITE Present_Value = (ValidDataTypeValue: Any value of a datatype that requires no coercion)
4. WAIT (**Channel Write Fail Time** * LEN)
5. VERIFY N = ValidDataTypeValue
6. VERIFY Write_Status = SUCCESSFUL

7.3.2.X40.12 Write Priority Test

Purpose: To check that the Channel object uses a priority level specified by a write service when the Channel object propagate its Present_Value to (a) target object property reference(s). If no priority level is specified, check that 16 is used by default.

Test Concept: When a valid data type value is written to a Present_Value of the Channel object by a WriteProperty request and a 'Priority' is provided in the write, the Channel object will use this same priority to command the referenced properties. When another value is written to a Present_Value of the Channel object by a WriteProperty request with no 'Priority' specified, the Channel object will use a priority 16 by default to command the referenced properties.

Configuration Requirements: Configure entry X of the Channel object's List_Of_Object_Property_References to refer to a writable object property of a specific data type on the IUT such that no coercion will occur. Refer to the Table 12-63 – Datatype Coercion Rules from ASHRAE 135 for the data types that require no coercion. The referenced property must contain a Priority_Array property.

Test Steps:

```
1.  READ LEN = List_Of_Object_Property_References, ARRAY INDEX = 0
2.  READ N = List_Of_Object_Property_References, ARRAY INDEX = X
3.  TRANSMIT WriteProperty-Request,
        'Object Identifier' =       (Object Identifier of the Channel object)
        'Property Identifier'=      Present_Value
        'Property Value'=           (V1: Any value of a datatype that requires no coercion)
        'Priority' =                (P1: Any valid value but 16)
4.  WAIT (Channel Write Fail Time * LEN)
5.  RECEIVE BACnet-SimpleACK-PDU
6.  TRANSMIT ReadProperty-Request,
        'Object Identifier' =   (Object Identifier of N)
        'Property Identifier =  Priority_Array
        'Property Array Index' =    P1
7.  RECEIVE ReadProperty-ACK,
        'Object Identifier' =   (Object Identifier of N)
        'Property Identifier' = Priority_Array
        'Property Array Index' =    P1
        'Property Value' =      V1
8.   TRANSMIT WriteProperty-Request,
        'Object Identifier' =       (Object Identifier of the Channel object)
        'Property Identifier'=  Present_Value
        'Property Value'=           (V2: Any value of a datatype that requires no coercion)
9.   WAIT (Channel Write Fail Time * LEN)
10. TRANSMIT ReadProperty-Request,
        'Object Identifier' =   (Object Identifier of N)
        'Property Identifier =  Priority_Array
        'Property Array Index' =    16
11. RECEIVE ReadProperty-ACK,
        'Object Identifier' =   (Object Identifier of N)
        'Property Identifier' = Priority_Array
        'Property Array Index' =    16
        'Property Value' =      V2
```

7.3.2.X40.13 Writing with a NULL Value Test

Purpose: To check that the Channel object ignores datatype errors when writing a NULL value to a non-commandable target.

Test Concept: This test is to check a special exception of Write_Status reporting SUCCESSFUL after propagating a NULL value to both a commandable and non-commandable property. Writing a NULL value to a commandable property will result in a property relinquishing a value to a Relinquish_Default value. However, writing a NULL value to a non-commandable property will result in a property remaining a current value. If a non-commandable target property is on a remote device, the IUT will receive either an ERROR INVALID_DATATYPE or REJECT INVALID_PARAMETER_DATA_TYPE. The Write_Status after such events should report SUCCESSFUL instead of FAILED.

Configuration Requirements: Configure entry X of the Channel object's List_Of_Object_Property_References to refer to a commandable property that accepts NULL value and configure entry Y of the List_Of_Object_Property_References to a non-commandable property that rejects a NULL value with either ERROR INVALID_DATATYPE or REJECT INVALID_PARAMETER_DATA_TYPE. For a commandable property, all prioritized commands has to be relinquished and any minimum on/off time has to be accounted for prior to the test. An initial value of a commandable property, N1 must be different from a value of its Relinquish_Default. N1's Priority_Array has only one non-Null value in it and it is in the priority array level that the Channel object is targeting.

Test Steps:

```
1.  READ LEN = List_Of_Object_Property_References, ARRAY INDEX = 0
-- Read an initial value of target properties
2.  READ P1 = List_Of_Object_Property_References: ARRAY INDEX = X, which is a commandable property
3.  READ P2 = List_Of_Object_Property_References: ARRAY INDEX = Y, which is a non-commandable property

-- Find expected values of the target properties after a NULL value is written to them
4.  READ V1 = P1.Relinquish_Default
5.  READ V2 = P2

-- Make the Channel object to propagate NULL value to targets
6.  WRITE Present_Value  = NULL
7.  WAIT (Channel Write Fail Time * LEN)
8.  IF (P2 is on external) THEN
        RECEIVE WritePropertyMultiple-Error
            'Error Class' =     PROPERTY,
            'Error Code' =      INVALID_DATATYPE |
        RECEIVE BACnet-Reject-PDU
            'Reject Reason' =       INVALID_PARAMETER_DATATYPE

--Check that P1 has Relinquish_Default value and P2 remains the same
9.  VERIFY P1 = V1
10. VERIFY P2 = V2

--Check that the Channel object ignores the datatype error and Write_Status is SUCCESSFUL
11. VERIFY Write_Status =       SUCCESSFUL
```

BTL-15.2h-5: Alert Enrollment Object [BTLWG-69]

Overview:
Add support for Alert Enrollment testing

Changes:

[In BTL Checklist, add the following to the Object section]

Alert Enrollment Object		
	R	Base requirements
	R	Supports AE-N-I-B

Alarm and Event - Notification - Internal - B		
		...
	C[3]	Implements a *Standard or* Proprietary algorithm using Extended notifications
		...

[In BTL Test Plan, update section 1.1 External Document References to include addenda q]

135.1-2013q	*SSPC Test standard Addendum 135.1-2013q (This addendum is final and was approved by ASHRAE and by the American National Standards Institute on December 7, 2018.).* *This addenda updated alarm and event tests for protocol revisions 13 and higher.*

[In BTL Test Plan, Add the following clause]

3.52 Alert Enrollment Object

3.52.1 Base Requirements

Base requirements must be met by any IUT that can contain Alert Enrollment Objects.

135.1-2013q - 7.3.1.X1.1 - Alert Enrollment Reports The Source Object		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
135.1-2013q - 7.3.1.X1.2 Alert Enrollment Does Not Generate Acknowledgeable Transitions		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

[In BTL Test Plan, Replace entire existing sections 5.2.15, and 5.3.31 with the following]

5.2.15 Implements a Standard or Proprietary Algorithm Using Extended Notifications

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

BTL - 8.4.X6 - Extended Algorithm Tests (ConfirmedEventNotification)		
	Test Conditionality	Must be executed.
	Test Directives	

	Testing Hints	
BTL - 8.5.X6 - Extended Algorithm Tests (UnconfirmedEventNotification)		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

5.3.31 Implements a Proprietary Algorithm

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

BTL - 8.4.X6 - Extended Algorithm Tests (ConfirmedEventNotification)		
	Test Conditionality	Must be executed.
	Test Directives	This test shall be executed with an Event Enrollment object that is configured to monitor a property in a device other than the IUT.
	Testing Hints	
BTL - 8.5.X6 - Extended Algorithm Tests (UnconfirmedEventNotification)		
	Test Conditionality	Must be executed.
	Test Directives	This test shall be executed with an Event Enrollment object that is configured to monitor a property in a device other than the IUT.
	Testing Hints	

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

8.4.X6 EXTENDED Algorithm Tests (ConfirmedEventNotification)
Reason for Change: Addition of Alert Enrollment object in Protocol_Revision 13.

Purpose: To verify the correct generation EXTENDED event notifications.

Test Concept: The object begins the test in a NORMAL state. The IUT is manipulated in order to cause a transition of the object under test. The event notification content is verified for correctness.

Configuration Requirements: The IUT shall be configured such that the Event_Enable property has a value of TRUE for the transition type that will be generated. The Issue_Confirmed_Notifications property shall have a value of TRUE.

Test Steps:

```
1.  IF (the object generates TO-OFFNORMAL transitions) THEN
2.      READ ES1 = Event_State
3.      MAKE (an OFFNORMAL condition exist)
4.      WAIT (Time_Delay)
5.      BEFORE Notification Fail Time
            RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' =              (any valid process ID),
                'Initiating Device Identifier' =    IUT,
                'Event Object Identifier' =         (the event generating object),
                'Time Stamp' =                      (TS1: the current local time),
                'Notification Class' =              (the configured notification class),
                'Priority' =                        (the value configured for TO_OFFNORMAL),
                'Event Type' =                      EXTENDED,
                'Notify Type' =                     EVENT | ALARM,
                'AckRequired' =                     TRUE | FALSE,
                'From State' =                      ES1,
                'To State' =                        (ES2: any offnormal valid event state),
                'Event Values' =                    ( (any valid vendor id),
                                                      (any valid event-type),
                                                      (a list of 0 or more valid parameters)
                                                    )
```

```
6.        TRANSMIT BACnet-SimpleACK-PDU
7.        IF (the object being tested is NOT an Event Enrollment object) THEN
               VERIFY Status_Flags = (TRUE, FALSE,?,?)
8.        VERIFY Event_State = ES2
9.        VERIFY Event_Time_Stamps = (TS1, *, *)

10.  IF (the object generates TO_NORMAL transitions) THEN
11.       READ ES1 = Event_State
12.       MAKE (a NORMAL condition exist)
13.       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),
                    'Time Stamp' =                      (TS2: the current local time),
                    'Notification Class' =              (the configured notification class),
                    'Priority' =                        (the value configured to correspond to a
                                                        TO-NORMAL transition),
                    'Event Type' =                      EXTENDED,
                    'Notify Type' =                     EVENT | ALARM,
                    'AckRequired' =                     TRUE | FALSE,
                    'From State' =                      ES1,
                    'To State' =                        NORMAL,
                    'Event Values' =                    ( (any valid vendor id),
                                                          (any valid event-type),
                                                          (a list of 0 or more valid parameters)
                                                        )

14.       TRANSMIT BACnet-SimpleACK-PDU
15.       IF (the object being tested is NOT an Event Enrollment object) THEN
                    VERIFY Status_Flags = (FALSE, FALSE, ?, ?)
16.       VERIFY Event_State = NORMAL
17.       VERIFY Event_Time_Stamps = (*, *, TS2)
```

Notes to Tester: The 'Message Text' parameter is omitted in the test description because it is optional. The IUT may include this parameter in the notification messages. The time stamps indicated by "*" can have any valid value.

8.5.X6 EXTENDED Algorithm Tests (UnconfirmedEventNotification)
Reason for Change: Addition of Alert Enrollment object in Protocol_Revision 13.

Purpose: To verify the correct generation EXTENDED event notifications.

Configuration Requirements: The IUT shall be configured such that the Event_Enable property has a value of TRUE for the transition type that will be generated. The Issue_Confirmed_Notifications property shall have a value of FALSE.

Test Steps: The test steps for this test case are identical to the test steps in 8.4.X6 except that the event notification requests are UnconfirmedEventNotification requests and the TD does not acknowledge receiving the notifications.

BTL-15.2h-6: Alarm and Event - Configurable Recipient Lists [BTLWG-24]

Overview:

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

These changes adopt a completed SSPC addendum.

Changes:

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

Notification Class		
	R	Base Requirements
	C[1]~~BTL-R~~	Supports DM-DDB-A
	C[~~1~~*23*]	Supports writable Recipient_List properties
	C[~~1~~*2*]	Supports read-only Recipient_List properties
	BTL-C[3]	*Supports AE-CRL-B*
	[1] *Required if "Supports writable Recipient_List properties".* [~~1~~*2*] At least one of these options must be supported. [3] *Required if the IUT claims device profile B-AAC or B-BC.*	

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

Alarm and Event - Configurable Recipient Lists - B		
	R	Base Requirements
	BTL-R[1]	Supports DS-WP-B
	R	Supports DM-DDB-A
	[1] The Recipient_List properties of all Notification Class and Notification Forwarder objects present in the device shall be writable.	

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

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

5.X21.1 Base Requirements

There are no base requirements tests for this section. Existing tests for Notification Class object ensure Recipient_List supports writing all forms.

5.X21.2 Supports DS-WP-B

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

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims "Supports writable Recipient_List properties" in Notification Class objects, and if it supports Notification Forwarder objects claims "Supports writable Recipient_List properties" in Notification Forwarder objects.
	Testing Hints	

5.X21.3 Supports DM-DDB-A

The IUT supports DM-DDB-A.

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

BTL-15.2h-7: Data Sharing - WriteGroup Internal/External-B [BTLWG-296]

Reason for Change:

DS-WG-I-B and DS-WG-E-B (**Data Sharing-WriteGroup Internal/External B**), introduced in **Protocol Revision 14** need additional testing for Channel object added to **TP16**

Background:

- WriteGroup is used to facilitate the efficient distribution of values to a large number of devices and objects.
- The WriteGroup service is used by a sending BACnet-user to update arbitrary Channel objects' Present_Value properties for a particular numbered control group.
- WriteGroup service is an unconfirmed service using a unicast, multicast or broadcast address.

Relations with existing Channel Object Tests (BTLWG-32):

- The Channel object tests contain a single test for DS-WG-I-B, which is a REQUIRED test under Base Requirement section. **BTL-7.3.2.X.40.3 - WriteGroup Service Support Test**

- The device supporting Channel object must support execution of the WriteGroup service and claim DS-WG-I-B.

[In BTL Checklist under 4 - Data Sharing BIBBs, add a new section "**Data Sharing -WriteGroup-Internal - B" and "Data Sharing -WriteGroup-External - B**]

4 Data Sharing BIBBs

Support	Listing	Option
Data Sharing - WriteGroup - Internal - B		
	R	Base Requirements
	R	Supports the Channel object that can propagate values to local objects
	O	Supports the Channel object with a writable Allow_Group_Delay_Inhibit property
Data Sharing - WriteGroup - External - B		
	R	Base Requirements
	R	Supports DS-WG-I-B
	R	Supports DS-WP-A
	R	Supports the Channel Object that can propagate values to objects on remote devices

[In BTL Test Plan under 4 - Data Sharing BIBBs, add a new section "**Data Sharing-WriteGroup-Internal-B**" and "**Data Sharing -WriteGroup-External-B"**]

4.22 - WriteGroup - Internal - B

4.22.1 Base Requirements

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

There are no base requirements tests for this section.

4.22.2 Supports the Channel Object that can Propagate Values to Local Objects

The B device shall execute the WriteGroup service when appropriate using given primitive values. This will be confirmed using Channel objects.

BTL - 9.X40.1.X1 - Channel and Group Number Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 9.X40.1.X2 - Write Priority and Overriding Priority Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 9.X40.1.X3 - Relinquish Control Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

4.22.3 Supports the Channel Object with a Writable Allow_Group_Delay_Inhibit Property

The B device shall inhibit any execution delay when appropriate.

BTL - 9.X40.1.X4 - Inhibit Delay Test with WriteGroup		
	Test Conditionality	This test shall be skipped if the Channel object does not support at least 2 entries in the List_Of_Object_Property_References.
	Test Directives	
	Testing Hints	

4.23 - WriteGroup - External - B

4.23.1 Base Requirements

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

There are no base requirements tests for this section.

4.23.2 Supports DS-WG-I-B

The IUT shall support DS-WG-I-B in order to claim DS-WG-E-B

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DS-WG-I-B
	Testing Hints	

4.23.3 Supports DS-WP-A

The IUT shall support DS-WP-A in order to claim DS-WG-E-B

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

4.23.4 Supports the Channel Object that can propagate values to objects on remote devices

The B device shall execute the WriteGroup service when appropriate, and the Channel object can propagate values to remote devices by using either WriteProperty or WritePropertyMultiple request.

BTL - 7.3.2.X40.4 - Propagation Entirety Test		
	Test Conditionality	Must be executed.
	Test Directives	Write to the Channel Object by a WriteGroup. Use external object references for List_Of_Object_Property_References.
	Testing Hints	

[In BTL Specified Tests, add WriteGroup -B specific tests in section 9.X40]

9.X40.1.X1 Channel and Group Number Test

Purpose: To verity that the Channel object executes a WriteGroup service request only when containing a specified channel number and Group Number by a request, and the Channel object ignores a request otherwise. If a Group Number is 0, the Channel object ignores a service even when its Control_Groups property is set to 0.

Test Concept: The Channel Object, O1 will be assigned a specific value to its channel number and Group Number. When a device containing O1 receives a WriteGroup service, O1 executes the request and propagate a specified value to its destination only if A) the O1's channel number is the same as specified number by the request and B) the O1's Control_Groups contains the specified Group Number, except for a case when a Group Number was 0.

Configuration Requirements: Configure entry X of the Channel object's List_Of_Object_Property_References to refer to a commendable property of an object on either a local or remote device. For a commandable property, all prioritized commands has to be relinquished and any minimum on/off time has to be accounted for prior to the test. An initial value of a commandable property must be the same as RD, which is its Relinquish_Default value. The value to be propagated must be a valid value that does not require coercion.

Test Steps:

```
-- Obtain the data which will be used for the Channel Write Fail Time later in the steps
1.  READ LEN = List_Of_Object_Property_References, ARRAY INDEX = 0
-- Obtain the Channel Object's target object reference
2.  READ PR = List_Of_Object_Property_References, ARRAY INDEX = X
-- Set arbitrary numbers for Channel_Number and Control_Group to O1
3.  TRANSMIT WritePropertyMultiple-Request,
        'Object Identifier' =    O1,
        'Property Identifier' =  Channel_Number
        'Property Value' =       (CN: Any valid value)
        'Property Identifier' =  Control_Groups
        'Property Value' =       (CG: Any length of an array containing at least 1 non-zero element)
4.  RECEIVE BACnet-Simple-ACK-PDU
-- Send a WriteGroup with a mismatching channel number and Group Number
5.  TRANSMIT WriteGroup-Request,
        'Group Number' =     (A valid value larger than 0 and not contained by CG)
        'Write Priority' =   (any valid value),
        'Change List' =      (Any valid value different than CN, no overriding priority, Y: a valid value different than RD)
6.  WAIT Channel Write Fail Time * LEN
-- Make sure that O1 did not propagate a value to its target references
7.  VERIFY PR = RD
-- Send a WriteGroup with a matching channel number and a mismatching group number
8.  TRANSMIT WriteGroup-Request,
        'Group Number' =     (A valid value larger than 0 and not contained by CG)
        'Write Priority' =   (any valid value),
        'Change List' =      (CN, no overriding priority, Y: a valid value different than RD)
9.  WAIT Channel Write Fail Time * LEN
-- Make sure that O1 did not propagate value to its target references
10. VERIFY PR = RD
-- Send a WriteGroup with a mismatching channel number and a matching group number
```

11. TRANSMIT WriteGroup-Request,
    'Group Number' = (Any non 0 values contained by CG)
    'Write Priority' = (any valid value),
    'Change List' = (Any valid value different than CN, no overriding priority, Y: a valid value different than RD)
12. WAIT **Channel Write Fail Time** * LEN

-- Make sure that O1 did not propagate value to its target references

13. VERIFY PR = RD

-- Send a WriteGroup service with a matching channel number and group number

14. TRANSMIT WriteGroup-Request,
    'Group Number' = (Any non 0 values contained by CG)
    'Write Priority' = (any valid value),
    'Change List' = (CN, no overriding priority, Y: a valid value different than RD)

-- Make sure that O1 did propagate value to its target references

15. VERIFY PR = Y

-- Change Control_Groups to 0

16. TRANSMIT WritePropertyMultiple-Request,
    'Object Identifier' = O1,
    'Property Identifier' = Control_Groups
    'Property Value' = 0
17. RECEIVE BACnet-Simple-ACK-PDU

-- Send a WriteGroup with 0 Group number

18. TRANSMIT WriteGroup-Request,
    'Group Number' = 0
    'Write Priority' = (any valid value),
    'Change List' = (CN, no overriding priority, Z: a valid value different than Y)
19. WAIT **Channel Write Fail Time** * LEN

-- Make sure that O1 did not propagate value to its target references

20. VERIFY PR = Y

9.X40.1.X2 Write Priority and Overriding Priority Test

Purpose: To verity that the overridingPriority, if provided, specifies the priority for writing the value. Otherwise the 'Write Priority' parameter specifies the priority for writing.

Test Concept: The Channel Object, O1 receives the WriteGroup with P1 as its 'Write Priority' and it is verified that P1 is used for writing the value. O1 then receives another WriteGroup with P1 as its 'Write Priority' and P2 as its overridingPriority and it is verified that P2 is used for writing the value.

Configuration Requirements: Configure one of the entry for the Channel object's List_Of_Object_Property_References to refer to a commendable property of an object O2 with Priority_Array on either a local or remote device. All prioritized commands has to be relinquished and any minimum on/off time has to be accounted for prior to the test.

Test Steps:

-- Obtain the data which will be used for the Channel Write Fail Time later in the steps

1. READ LEN = List_Of_Object_Property_References, ARRAY INDEX = 0

-- Write to O1 using P1 as its Write Priority

2. TRANSMIT WriteGroup-Request,
    'Group Number' = (one of the Control_Group values configured in O1),
    'Write Priority' = (P1: Any valid value)
    'Change List' = (O1's channel number, no overriding priority, Y: any valid value)
3. WAIT **Channel Write Fail Time** * LEN

-- Make sure that P1 is used for writing the value

4. VERIFY (O2), Priority_Array = Y, Array_Index = P1

-- Write to O1 using P1 as its Write Priority, P2 as its overridingPriority

5. TRANSMIT WriteGroup-Request,
    'Group Number' = (one of the Control_Group values configured in O1),
    'Write Priority' = P1
    'Change List' = (O1's channel number, P2: Any valid value different than P1, Z: any valid value different than Y)
6. WAIT **Channel Write Fail Time** * LEN

-- Make sure that P2 is used for writing the value

7. VERIFY (O2), Priority_Array = Z, Array_Index = P2

-- Make sure that no update on Priority_Array[P1]
8. VERIFY (O2), Priority_Array = Y, Array_Index = P1

9.X40.1.X3 Relinquish Control Test

Purpose: To verify that if a BACnetGroupChannelValue specifies a NULL value, it serves the same function as if NULL had been used with WriteProperty.

Test Concept: The Channel Object, O1 receives the WriteGroup service to propagate a value to its destination object property reference, PR. PR is verified to have the value updated accordingly. The O1 then receives another WriteGroup service with the BACnetGroupChannelValue specifying a NULL value. PR is verified to have its Relinquish_Default value.

Configuration Requirements: Configure entry X of the Channel object's List_Of_Object_Property_References to refer to a commendable property of an object O2 with a Relinquish_Default set to RD on either a local or remote device. All prioritized commands has to be relinquished and any minimum on/off time has to be accounted for prior to the test.

Test Steps:
-- Obtain the data which will be used for the Channel Write Fail Time later in the steps
1. READ LEN = List_Of_Object_Property_References, ARRAY INDEX = 0
-- Obtain the Channel Object's target object reference
2. READ PR = List_Of_Object_Property_References, ARRAY INDEX = X
-- Let the Channel Object propagate a value to its target
3. TRANSMIT WriteGroup-Request,
    'Group Number' = (One of the Control_Group values configured in O1),
    'Write Priority' = (Any valid value)
    'Change List' = (O1's channel number, no overriding priority, X: any valid value different than RD)
4. WAIT **Channel Write Fail Time** * LEN
5. VERIFY PR = X
-- Let the Channel Object relinquish control of the target
6. TRANSMIT WriteGroup-Request,
    'Group Number' = (One of the Control_Group values configured in O1),
    'Write Priority' = (Any valid value)
    'Change List' = (O1's channel number, no overriding priority, NULL)
7. WAIT **Channel Write Fail Time** * LEN
8. VERIFY PR = RD

9.X40.1.X4 Inhibit Delay Test with WriteGroup

Purpose: In the case of WriteGroup, verify that Execution_Delay always occurs unless the WriteGroup service parameter 'Inhibit Delay' is TRUE, and the Channel object property Allow_Group_Delay_Inhibit is present and has the value TRUE.

Test Concept: Setup List_Of_Object_Property_References of the Channel Object, O1 to contain 2 valid entries PR1, and PR2 and provide each with an execution delay (ED1 and ED2) which are larger than 0. Allow_Group_Delay_Inhibit is set to TRUE. When a WriteGroup service is sent to O1 without 'Inhibit Delay' parameter, it is verified that Execution_Delay occurs. When another WriteGroup service is sent to O1 with 'Inhibit Delay' set to False, it is verified that Execution_Delay still occurs. Finally, when another WriteGroup service is sent with 'Inhibit Delay' set to TRUE, it is verified that Execution_Delay does not occur.

Configuration Requirements: PR1 and PR2 shall be references to writable properties and shall be the same datatype. ED1 and ED2 shall be values which are large enough that the delay between writes is sufficient for the test. V1 and V2 shall be of the expected datatype for PR1 and PR2 so that no coercion occur, and shall be different values.

-- Setup the Channel object

1. WRITE List_Of_Object_Property_References = (PR1, PR2)
2. WRITE Execution_Delay = (ED1, ED2)
3. WRITE Allow_Group_Delay_Inhibit = TRUE
4. READ V1 = PR1
5. READ V2 = PR2

```
-- Send a WriteGroup without 'Inhibit Delay' parameter
6.  TRANSMIT WriteGroup-Request,
        'Group Number' =    (one of the Control_Group values configured in O1),
        'Write Priority' =  (Any valid value)
        'Change List' =     (O1's channel number, no overriding priority, X: any valid value different than V1 or V2)
7.  WAIT Channel Write Fail Time
-- Make sure that Execution_Delay occurs
8.  VERIFY PR1= V1
9.  VERIFY PR2 = V2
10. WAIT (ED1)
11. VERIFY PR1 = X
12. VERIFY PR2 = V2
13. WAIT (ED2 – ED1)
14. VERIFY PR2 = X
-- Send a WriteGroup with 'Inhibit Delay' set to FALSE
15. TRANSMIT WriteGroup-Request,
        'Group Number' =    (one of the Control_Group values configured in O1),
        'Write Priority' =  (Any valid value)
        'Change List' =     (O1's channel number, no overriding priority, Y: any valid value different from X)
        'Inhibit Delay' =           FALSE
16. WAIT Channel Write Fail Time
-- Make sure that Execution_Delay occurs
17. VERIFY PR1= X
18. VERIFY PR2 = X
19. WAIT (ED1)
20. VERIFY PR1 = Y
21.  VERIFY PR2 = X
22. WAIT (ED2 – ED1)
23. VERIFY PR2 = Y
-- Send a WriteGroup with 'Inhibit Delay' set to TRUE
24. TRANSMIT WriteGroup-Request,
        'Group Number' =    (one of the Control_Group values configured in O1),
        'Write Priority' =  (Any valid value)
        'Change List' =     (O1's channel number, no overriding priority, Z: any valid value different from Y)
        'Inhibit Delay' =           TRUE
-- Make sure that Execution_Delay does NOT occurs
25. WAIT Channel Write Fail Time
26. VERIFY PR1= Z
27. VERIFY PR2 = Z
```