BACnet® TESTING LABORATORIES
ADDENDA

Addendum k to BTL Test Package 15.2

Revision 4
Revised 03/22/2019

Approved by the BTL Working Group on 24-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-15.2k-1: Update Lifetime Test [BTLWG-87]

Overview:

CR-0369 clarified that the testing in BTL - 9.10.2.X3 is for a testable point only when the supplied 'Lifetime' parameter in the SubscribeCOV-Request is less than the MAX_UNSIGNED supported.

Typographic conventions: The modification of this test is shown with ~~strike-through of the removed material~~ and *italic expression of new material*.

Changes:

[In BTL Specified Tests, modify existing test, as indicated]

9.10.2.X3 The Lifetime Parameter is Out of Range

Reason for Change: 135-2008h.5. ~~Modified to relax allowed rejection~~ *Mandates a specific* response *in this situation*. Also CR-0369 clarified that the testing shall only supply a 'Lifetime' parameter in the SubscribeCOV-Request less than the MAX_UNSIGNED supported.

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

Test Conditionality: If the device cannot be observed to produce a Result(-) result when the MAX_UNSIGNED32 (4294967295 approx. 136 years) is the supplied 'Lifetime' parameter sent in the subscription, then this test may be skipped.

Test Steps:

```
1.  TRANSMIT SubscribeCOV-Request,
        'Subscriber Process Identifier' = (any valid process identifier),
        'Monitored Object Identifier' = (any object in the IUT that supports COV),
        'Issue Confirmed Notifications' = TRUE,
        'Lifetime' = (a value *less than the MAX_UNSIGNED supported by the IUT, but large enough to produce a Result(-)
result* ~~larger than that supported~~ by the IUT)
2.  IF (Protocol_Revision is present and Protocol_Revision >= 10) THEN
        RECEIVE BACnet-Error-PDU,
            Error Class = SERVICES,
            Error Code = VALUE_OUT_OF_RANGE
    ELSE
        RECEIVE BACnet-Error-PDU,
            Error Class = SERVICES,
            Error Code = VALUE_OUT_OF_RANGE | SERVICE_REQUEST_DENIED | OTHER
                         | (RECEIVE BACnet-Reject-PDU,
            'Reject Reason' = PARAMETER_OUT_OF_RANGE)
```

BTL-15.2k-2: Add WritePropertyMultiple Array Resize Testing [BTLWG-111]

Overview:

The existing test plan has a test case using WriteProperty service.
We have added a new test case using WritePropertyMultiple service.

The test in this clause shall be applied to resizable arrays in devices claiming Protocol_Revision 4 or higher. It may be applied to resizable arrays in devices claiming Protocol_Revision 3 or lower, but only where conformance to the rules on resizing arrays of Protocol_Revision 4 is claimed.

Typographic conventions: This new testcase and content is simply shown without highlight in normal font. The correction is highlighted and strike off.

Changes:

[Change in Test Plan section 4.8, Data Sharing - WritePropertyMultiple – B modified the Test ID, Title, Configuration and Test Directives.]

4.8 Data Sharing-WritePropertyMultiple - B

4.8.7 Contains Resizable Array Properties

The IUT contains, or can be made to contain, an array property that is resizable by writing to the 0th element.

…		
BTL - 7.3.1.X16 - Array Resizing Test using WritePropertyMultiple service		
	Test Conditionality	This test shall be executed if the IUT is protocol revision 4 or higher..
	Test Directives	Execute on at least one instance of each resizable array property, both standard and proprietary
	Testing Hints	

[Add this new test in BTL Specified Tests, as shown]

7.3.1.X16 Array Resizing Test using WritePropertyMultiple service
Reason For Change: The existing test plan has a test case using WriteProperty service. We have added a new test case using WritePropertyMultiple service.

Purpose: To verify that resizable arrays are resized in accordance with the rules added in Protocol_Revision 4.

Test Concept: The resizable array property P1 of object O1 is written with WritePropertyMultiple as a whole to set it to a non-zero size. It is then resized smaller and larger by writing the entire array. It is then resized smaller and larger by writing to element number zero. An attempt is made to increase it with an invalid write. After each operation, the array size and array contents are checked. Finally, if it can be resized to have zero elements, it is then written to size zero. If possible, all elements in the arrays should be distinguishable from each other and across WritePropertyMultiple operations.

Test Steps:

```
1.  TRANSMIT WritePropertyMultiple-Request,
        'Object Identifier' = O1,
        'Property Identifier' = P1,
        'Property Value' = (array A1 of non-zero size N1)
2.  RECEIVE BACnet-SimpleACK-PDU
3.  VERIFY P1= (array A1), ARRAY INDEX = 0, (array size i.e. N1)

--Resize the array to make it smaller in size

4.  TRANSMIT WritePropertyMultiple-Request,
        'Object Identifier' = O1,
        'Property Identifier' = P1
        'Property Value' = (array A2 of non-zero size N2, where N2 ≤ N1)
```

```
5.  RECEIVE BACnet-SimpleACK-PDU
6.  VERIFY P1 = (array A2), ARRAY INDEX = 0, (array size N2)

--Resize the array to make it larger in size

7.  TRANSMIT WritePropertyMultiple-Request,
        'Object Identifier' = O1,
        'Property Identifier' = P1
        'Property Value' = (array A3 of non-zero size N3, where N3 ≥ N1),
8.  RECEIVE BACnet-SimpleACK-PDU
9.  VERIFY P1 = (array A3), ARRAY INDEX = 0, (array size N3)

--Modify the existing content of element

10. TRANSMIT WritePropertyMultiple-Request,
        'Object Identifier' = O1,
        'Property Identifier' = P1,
        'Property Value' = (array A4 of non-zero unsigned value N4, where N4 ≤ N1),
11. RECEIVE BACnet-SimpleACK-PDU
12. VERIFY P= (array A4), ARRAY INDEX = 0, (array size N4)

--Resize the array by writing the size of the array

13. TRANSMIT WritePropertyMultiple-Request
        'Object Identifier' = O1,
        'Property Identifier' = P1
        'Property Value' = (N5, where N5 ≥ N4),
        'Property Array Index' = 0,
14. RECEIVE BACnet-SimpleACK-PDU
15. VERIFY (array contains unchanged first N4 elements of the array written in step 10, plus N5-N4 additional elements,
initialized to particular values for the array property being tested)
16. VERIFY P1, ARRAY INDEX = 0, (array size N5)

--Try to add the array element at Array Index which is greater than the size of the array

17. TRANSMIT WritePropertyMultiple-Request,
        'Object Identifier' = O1,
        'Property Identifier' = P1,
        'Property Value' = (one array element),
        'Property Array Index' = (N6, where N6 ≥ N5),
18. RECEIVE WritePropertyMultiple-Error
        'Error Class' = PROPERTY,
        'Error Code' = INVALID_ARRAY_INDEX
        'Object Identifier' = O1,
        'Property Identifier' = P1,
        'Property Array Index' = N6
19. VERIFY (array is unchanged from step 15)

--Resize the array to size zero

20. IF (the array can be resized to have zero elements) THEN
        TRANSMIT WritePropertyMultiple-Request,
            'Object Identifier' = O1,
            'Property Identifier' = P1,
            'Property Value' = (empty array),
        BACnet-SimpleACK-PDU
21. VERIFY P1 = (array is empty), ARRAY INDEX = 0, (array size is zero)
```

BTL-15.2k-3: Errata Changes for COV-A and COV-B Test Plan [BTLWG-165]

Overview:

Update the COV-A and COV-B Test Plan.

- Use consistent preamble and Test Directive terminology for the relationship of object types and data types.
- Update test names.

Changes:

[In BTL Test Plan, update sections below as specified.]

4.9 Data Sharing - Change Of Value - A

4.9.5 Can Subscribe for COV from Analog Objects

The IUT can subscribe for, receive, and process Change of Value notifications from *at least one* ~~Analog Input, Analog Output, and Analog Value~~ objects *using type REAL in its parameters*.

135.1-2013 - 9.2.1.1 - Change of Value Notifications		
	Test Conditionality	Either 9.2.1.1 or 9.3.2 must be executed.
	Test Directives	*Test one instance of each object type where Present_Value is of data type REAL.*
	Testing Hints	
135.1-2013 - 9.3.2 - Change of Value Notifications		
	Test Conditionality	Either 9.2.1.1 or 9.3.2 must be executed.
	Test Directives	*Test one instance of each object type where Present_Value is of data type REAL.*
	Testing Hints	

4.9.6 Can Subscribe for COV from Binary Objects

The IUT can subscribe for, receive, and process Change of Value notifications from *at least one* ~~Binary Input, Binary Output, and Binary Value~~ object ~~objects~~ *using type BACnetBinaryPV in its parameters*.

135.1-2013 - 9.2.1.1 - Change of Value Notifications		
	Test Conditionality	Either 9.2.1.1 or 9.3.2 must be executed.
	Test Directives	*Test one instance of each object type where Present_Value is of data type BACnetBinaryPV.*
	Testing Hints	
135.1-2013 - 9.3.2 - Change of Value Notifications		
	Test Conditionality	Either 9.2.1.1 or 9.3.2 must be executed.
	Test Directives	*Test one instance of each object type where Present_Value is of data type BACnetBinaryPV.*
	Testing Hints	

4.9.7 Can Subscribe for COV from Life Safety Objects

The IUT can subscribe for, receive, and process Change of Value notifications from *at least one* Life Safety Point *or* ~~and~~ Life Safety Zone object~~s~~ *using type BACnetLifeSafetyState in its parameters*.

135.1-2013 - 9.2.1.1 - Change of Value Notifications		
	Test Conditionality	Either 9.2.1.1 or 9.3.2 must be executed.
	Test Directives	*Test one instance of each object type where Present_Value is of data type BACnetLifeSafetyState.*
	Testing Hints	
135.1-2013 - 9.3.2 - Change of Value Notifications		
	Test Conditionality	Either 9.2.1.1 or 9.3.2 must be executed.
	Test Directives	*Test one instance of each object type where Present_Value is of data type BACnetLifeSafetyState.*
	Testing Hints	

…

4.9.9 Can Subscribe for COV from Multi-state Objects

The IUT can subscribe for, receive, and process Change of Value notifications from *at least one* ~~Multi-state Input, Multi-state Output and Multi-state Value~~ objects *using type Unsigned in its parameters*.

135.1-2013 - 9.2.1.1 - Change of Value Notifications		
	Test Conditionality	Either 9.2.1.1 or 9.3.2 must be executed.
	Test Directives	*Test one instance of each object type where Present_Value is of data type Unsigned.*
	Testing Hints	
135.1-2013 - 9.3.2 - Change of Value Notifications		
	Test Conditionality	Either 9.2.1.1 or 9.3.2 must be executed.
	Test Directives	*Test one instance of each object type where Present_Value is of data type Unsigned.*
	Testing Hints	

4.9.10 Can Subscribe for COV from CharacterString Objects

The IUT supports change of value notifications for at least one object using ~~of~~ type CharacterString *in its parameters*.

135.1-2013 - 9.2.1.1 - Change of Value Notifications		
	Test Conditionality	Must be executed.
	Test Directives	Test one instance of each object type where Present_Value is of data type CharacterString.
	Testing Hints	
135.1-2013 - 9.3.2 - Change of Value Notifications		
	Test Conditionality	Must be executed.
	Test Directives	Test one instance of each object type where Present_Value is of data type CharacterString.
	Testing Hints	

4.9.11 Can Subscribe for COV from Date Objects

The IUT supports change of value notifications for at least one object *using* ~~of~~ type Date *in its parameters*.

135.1-2013 - 9.2.1.1 - Change of Value Notifications		
	Test Conditionality	Must be executed.
	Test Directives	Test one instance of each object type where Present_Value is of data type Date.
	Testing Hints	
135.1-2013 - 9.3.2 - Change of Value Notifications		
	Test Conditionality	Must be executed.
	Test Directives	Test one instance of each object type where Present_Value is of data type Date.
	Testing Hints	

4.9.12 Can Subscribe for COV from DateTime Objects

The IUT supports change of value notifications for at least one object using ~~of~~ type *BACnet*DateTime *in its parameters* ~~Value~~.

135.1-2013 - 9.2.1.1 - Change of Value Notifications		
	Test Conditionality	Must be executed.
	Test Directives	Test one instance of each object type where Present_Value is of data type *BACnet*DateTime.
	Testing Hints	
135.1-2013 - 9.3.2 - Change of Value Notifications		
	Test Conditionality	Must be executed.
	Test Directives	Test one instance of each object type where Present_Value is of data type *BACnet*DateTime.

	Testing Hints	

4.9.13 Can Subscribe for COV from Integer Objects

The IUT supports change of value notifications for at least one object *using* ~~of~~ type Integer *in its parameters*.

135.1-2013 - 9.2.1.1 - Change of Value Notifications		
	Test Conditionality	Must be executed.
	Test Directives	Test one instance of each object type where Present_Value is of data type Integer.
	Testing Hints	
135.1-2013 - 9.3.2 - Change of Value Notifications		
	Test Conditionality	Must be executed.
	Test Directives	Test one instance of each object type where Present_Value is of data type Integer.
	Testing Hints	

4.9.16 Can Subscribe for COV from Positive Integer Objects

The IUT supports change of value notifications for at least one object of type Positive Integer.

135.1-2013 - 9.2.1.1 - Change of Value Notifications		
	Test Conditionality	Must be executed.
	Test Directives	Test one instance of each object type where Present_Value is *using* ~~of~~ data type *Unsigned, including the possible value zero* ~~Positive Integer~~.
	Testing Hints	
135.1-2013 - 9.3.2 - Change of Value Notifications		
	Test Conditionality	Must be executed.
	Test Directives	Test one instance of each object type where Present_Value is *using* ~~of~~ data type *Unsigned, including the possible value zero* ~~Positive Integer~~.
	Testing Hints	

[In BTL Test Plan, Update the names of tests in the Test Plan.]

4.10 Data Sharing - Change Of Value - B

BTL - 8.2.1 - Change of Value Notification from an Analog ~~Input, Analog Output, and Analog Value~~*~~, Large Analog Value, Integer Value, and Positive Integer Value~~* Object Present_Value Property

BTL - 8.2.2 - Change of Value Notification from an Analog ~~Input, Analog Output, and Analog Value~~*~~, Large Analog Value, Integer Value, and Positive Integer Value~~* Object Status_Flags Property

BTL - 8.2.3 - Change of Value Notification from a Binary ~~Input, Binary Output, and Binary Value~~ Object Present_Value Property

BTL - 8.2.4 - Change of Value Notification from a Binary ~~Input, Binary Output, and Binary Value~~ Object Status_Flags Property

BTL - 8.3.1 - Change of Value Notification from an Analog ~~Input, Analog Output, and Analog Value~~*~~, Large Analog Value, Integer Value, and Positive Integer Value~~* Object Present_Value Property

BTL - 8.3.2 - Change of Value Notification from an Analog ~~Input, Analog Output, and Analog Value~~*~~, Large Analog Value, Integer Value, and Positive Integer Value~~* Object Status_Flags Property

135.1-2013 - 8.3.3 - Change of Value Notification from a Binary Input, Binary Output, and Binary Value Object Present_Value Property

135.1-2013 - 8.3.4 - Change of Value Notification from a Binary ~~Input, Binary Output, and Binary Value~~ Object Status_Flags Property

BTL - 8.2.5 - Change of Value Notification from a ~~Multi-state Input, Multi-state Output, Multi-state Value, Life Safety Point, and Life Safety Zone~~*~~, CharacterString Value, OctetString Value, Date Value, Date Pattern Value, DateTime Value, DateTime Pattern Value, Time Value, or Time Pattern Value~~* *Discrete Valued* **Object Present_Value Property**

BTL - 8.2.6 - Change of Value Notification from a ~~Multi-state Input, Multi-state Output, Multi-state Value, Life Safety Point, and Life Safety Zone~~*~~, CharacterString Value, OctetString Value, Date Value, Date Pattern Value, DateTime Value, DateTime Pattern Value, Time Value, or Time Pattern Value~~* *Discrete Valued* **Object Status_Flags Property**

BTL - 8.3.5 - Change of Value Notification from a ~~Multi-state Input, Multi-state Output, Multi-state Value, Life Safety Point, and Life Safety Zone~~*~~, CharacterString Value, OctetString Value, Date Value, Date Pattern Value, DateTime Value, DateTime Pattern Value, Time Value, or Time Pattern Value~~* *Discrete Valued* **Object Present_Value Property**

BTL - 8.3.6 - Change of Value Notification from a ~~Multi-state Input, Multi-state Output, Multi-state Value, Life Safety Point, and Life Safety Zone~~*~~, CharacterString Value, OctetString Value, Date Value, Date Pattern Value, DateTime Value, DateTime Pattern Value, Time Value, or Time Pattern Value~~* *Discrete Valued* **Object Status_Flags Property**

BTL-15.2k-4: Add Testing for CHANGE_OF_DISCRETE_VALUE Algorithm [BTLWG-272]

Overview:

No test exists for this functionality. New algorithm for Protocol_Revision 16

Changes:

[In BTL Specified Tests, add new tests]

8.4.X9 CHANGE_OF_DISCRETE_VALUE Test (ConfirmedEventNotification)

Purpose: To verify correct operation of the CHANGE_OF_DISCRETE_VALUE event algorithm. This test applies to Event Enrollment objects with an Event_Type of CHANGE_OF_DISCRETE_VALUE.

Test Concept: pMonitoredValue is changed to a value different from the initial value. After pTimeDelay (pTimeDelay is the value for Time_Delay specified in the EventParameters), a TO-NORMAL transition occurs and a ConfirmedEventNotification is generated by the IUTl.

Configuration Requirements: An Event_Enrollment, EE1 is configured with an Object_Property_Reference, (O1, P1), such that P1 is of one of the following datatypes: BOOLEAN, Unsigned, Integer, Enumerated, CharacterString, Octet String, Date, Time, BACnetObjectIdentifier, or BACnetDateTime. Event_Enable is configured with a value of (T,T,T) and Event_Algorithm_Inhibit = FALSE. The Event_Parameters are configured with an Event Algorithm of CHANGE_OF_DISCRETE_VALUE and a value for Time_Delay that is within the allowable range for the IUT. The configured notification class is configured to send confirmed notifications to the TD. EE1 shall have an Event_State of NORMAL at the start of the test.

Test Steps:

```
1.  VERIFY Event_State = Normal
2.  MAKE (the referenced property have a value x: x differs from the initial value)
3.  WAIT (pTimeDelay)
4.  BEFORE Notification Fail Time
        RECEIVE ConfirmedEventNotification
            'Process Identifier' =            (any valid process ID),
            'Initiating Device Identifier' =  IUT,
            'Event Object Identifier' =       EE1,
            'Time Stamp' =                    (the current local datetime or time or sequence number),
            'Notification Class' =  (the notification class configured for EE1),
            'Priority' =   (the value configured to correspond to TO-NORMAL),
            'Event Type' =          CHANGE_OF_DISCRETE_VALUE,
            'Message Text' =        (optional, any valid message text),
            'Notify Type' =         ALARM | EVENT,
            'Ack Required' =        TRUE | FALSE,
            'From State' =          NORMAL,
            'To State' =   NORMAL,
            'Event Values' =        (x, Status_Flags of O1)
5.  TRANSMIT BACnet SimpleAck-PDU
6.  VERIFY Event_State = Normal
```

8.5.X10 CHANGE_OF_DISCRETE_VALUE Test (UnconfirmedEventNotification)

Purpose: To verify correct operation of the CHANGE_OF_DISCRETE_VALUE event algorithm. This test applies to Event Enrollment objects with an Event_Type of CHANGE_OF_DISCRETE_VALUE.

Test Concept: pMonitoredValue is changed to a value different from the initial value. After pTimeDelayNormal (pTimeDelay is the value for Time_Delay specified in the EventParameters), a TO-NORMAL transition occurs and a UnconfirmedEventNotification is generated by the IUTl.

Configuration Requirements: An Event_Enrollment, EE1 is configured with an Object_Property_Reference, (O1, P1), such that P1 is of one of the following datatypes: BOOLEAN, Unsigned, Integer, Enumerated, CharacterString, Octet String, Date, Time, BACnetObjectIdentifier, or BACnetDateTime. Event_Enable is configured with a value of (T,T,T) and Event_Algorithm_Inhibit = FALSE. The Event_Parameters are configured with an Event Algorithm of CHANGE_OF_DISCRETE_VALUE and a value for Time_Delay that is within the allowable range for the IUT. The configured notification class is configured to send unconfirmed notifications to the TD. EE1 shall have an Event_State of NORMAL at the start of the test.

Test Steps:

```
1.  VERIFY Event_State = Normal
2.  MAKE (the referenced property have a value x: x differs from the initial value)
3.  WAIT (pTimeDelay)
4.  BEFORE Notification Fail Time
        RECEIVE UnconfirmedEventNotification
            'Process Identifier' =          (any valid process ID),
            'Initiating Device Identifier' = IUT,
            'Event Object Identifier' =     EE1,
            'Time Stamp' =          (the current local datetime or time or sequence number),
            'Notification Class' =  (the notification class configured for EE1),
            'Priority' =    (the value configured to correspond to TO-NORMAL),
            'Event Type' =      CHANGE_OF_DISCRETE_VALUE,
            'Message Text' =  (optional, any valid message text),
            'Notify Type' =     ALARM | EVENT,
            'Ack Required' =  TRUE | FALSE,
            'From State' =      NORMAL,
            'To State' =    NORMAL,
            'Event Values' =  (x, Status_Flags of O1)
5.  VERIFY Event_State = Normal
```

[In BTL Test Plan 15.1, add the following entries for AE-N-I-B and AE-N-E-B]

5.2.X Implements the CHANGE_OF_DISCRETE_VALUE Algorithm

The IUT contains, or can be made to contain, an object that can generate EventNotifications with an Event_Type of CHANGE_OF_DISCRETE_VALUE.

BTL - 8.4.X9 - CHANGE_OF_DISCRETE_VALUE Test (ConfirmedEventNotification)		
	Test Conditionality	Must be executed.
	Test Directives	This test must be repeated for each datatype, BOOLEAN, Unsigned, Integer, Enumerated, CharacterString, Octet String, Date, Time, BACnetObjectIdentifier, or BACnetDateTime supported by the Object_Property_Reference property of the Event Enrollment object.
	Testing Hints	
BTL - 8.5.X10 - CHANGE_OF_DISCRETE_VALUE Test (UnconfirmedEventNotification)		
	Test Conditionality	Must be executed.
	Test Directives	This test must be repeated for each datatype, BOOLEAN, Unsigned, Integer, Enumerated, CharacterString, Octet String, Date, Time, BACnetObjectIdentifier, or BACnetDateTime supported by the Object_Property_Reference property of the Event Enrollment object.
	Testing Hints	

5.3.X Implements the CHANGE_OF_DISCRETE_VALUE Algorithm

The IUT contains, or can be made to contain, an object that can generate EventNotifications with an Event_Type of CHANGE_OF_DISCRETE_VALUE.

BTL - 8.4.X9 - CHANGE_OF_DISCRETE_VALUE Test (ConfirmedEventNotification)		
	Test Conditionality	Must be executed.

	Test Directives	This test must be repeated for each datatype, BOOLEAN, Unsigned, Integer, Enumerated, CharacterString, Octet String, Date, Time, BACnetObjectIdentifier, or BACnetDateTime supported by the Object_Property_Reference property of the Event Enrollment object.
	Testing Hints	
BTL - 8.5.X10 - CHANGE_OF_DISCRETE_VALUE Test (UnconfirmedEventNotification)		
	Test Conditionality	Must be executed.
	Test Directives	This test must be repeated for each datatype, BOOLEAN, Unsigned, Integer, Enumerated, CharacterString, Octet String, Date, Time, BACnetObjectIdentifier, or BACnetDateTime supported by the Object_Property_Reference property of the Event Enrollment object.
	Testing Hints	

[In BTL Checklist 15.1.final, change the footnotes for the CHANGE_OF_DISCRETE_VALUE entry for AE-N-I-B and AE-N-E-B]

5 Alarm and Event Management BIBBs

Support	**Listing**	**Option**
Alarm and Event Management - Notification - Internal - B		
		…
	C[3,~~5~~,6]	Implements the CHANGE_OF_DISCRETE_VALUE algorithm
		…
		[1] Required if EventNotifications with service parameter AckRequired = True can be issued. [2] At least one of these options must be supported to claim support for this BIBB. [3] At least one of these options must be supported to claim support for this BIBB. It is recommended that a standard BACnet algorithm be used instead of a proprietary algorithm whenever possible. [4] At least one of these options must be supported to claim support for this BIBB. The BACnetDateTime form of the timestamp is the recommended option. [5]Contact BTL for interim tests for this algorithm. [6]Protocol_Revision 16 or higher must be claimed. [7]Protocol_Revision 17 or higher must be claimed. [8]Protocol_Revision 18 or higher must be claimed.

Support	**Listing**	**Option**
Alarm and Event Management - Notification - External - B		
		…
	C[1,~~2~~,3]	Implements the CHANGE_OF_DISCRETE_VALUE algorithm
		…
		[1] One of these options must be supported to claim support for this BIBB. It is recommended that a standard BACnet algorithm be used instead of a proprietary algorithm whenever possible. [2]Contact BTL for interim tests for this algorithm. [3]Protocol_Revision 16 or higher must be claimed. [4]Protocol_Revision 17 or higher must be claimed. [5]Protocol_Revision 18 or higher must be claimed.

BTL-15.2k-5: Add WritePropertyMultiple Negative Tests [BTLWG-411]

Overview:

Current test package does not have negative test cases for WritePropertyMultiple request to check error code with Property/Object Access Error for first element of 'List of Write Access Specifications' and that all writes after the first failed write attempt do not take place

Changes:

[In Test Plan sections 4.8.1 add as shown]

4.8 Data Sharing - WritePropertyMultiple - B

4.8.1 Base Requirements

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

…		
BTL - 9.23.2.X6 - Writing first element of 'List of Write Access Specifications' with a Property Access Error		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 9.23.2.X3 - Writing first element of 'List of Write Access Specifications' with Object Access Error		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 9.23.2.X4 - Writing first element of 'List of Write Access Specifications with a Write Access Error		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 9.23.2.X5 - WritePropertyMultiple Reject Test for first element of 'List of Write Access Specifications'		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision 10 or greater.
	Test Directives	
	Testing Hints	

[In BTL Specified Tests in section 9.23.2 add as shown]

9.23.2 Negative WritePropertyMultiple Service Execution Tests

9.23.2.X6 Writing first element of 'List of Write Access Specifications' with a Property Access Error

Purpose: To verify the ability to correctly execute a WritePropertyMultiple service request for which the first element of the 'List of Write Access Specifications' contains a specification for an unsupported property and all writes after the first failed write attempt do not take place.

Test Concept: An attempt is made to write to two properties in a single object. The first property is not supported for this object. The second property is supported for this object and writable. The objective is to verify that an appropriate error response is returned and that all writes after the first failed write attempt do not take place.

Configuration Requirements: If the IUT supports any writable scalar properties that are not commandable it shall be configured with one for use in this test. If no such properties are supported the IUT shall be configured with a writable array or commandable property and the test steps modified to account for this variation. In the test description O1 will be used to designate the object, P1 the unsupported property, and P2 the writable property having value X used.

Test Steps:
1. VERIFY (O1), P2 = X

2. TRANSMIT WritePropertyMultiple-Request,
   'Object Identifier' = O1,
   'Property Identifier' = P1,
   'Property Value' = (any valid value of the appropriate datatype for this property subject to the restrictions specified in the EPICS as defined in 4.4.2)
   'Property Identifier' = P2,
   'Property Value' = (any valid value not equal to X),
3. RECEIVE WritePropertyMultiple-Error,
   'Error Class' = PROPERTY,
   'Error Code' = UNKNOWN_PROPERTY,
   'Object Identifier' = O1,
   'Property Identifier' = P1
4. VERIFY (O1), P2 = X

9.23.2.X3 Writing first element of 'List of Write Access Specifications' with Object Access Error

Purpose: To verify the ability to correctly execute a WritePropertyMultiple service request for which the first element of the 'List of Write Access Specifications' contains a specification for an unsupported object and all writes after the first failed write attempt do not take place.

Test Concept: An attempt is made to write to a single property in two different objects. The first object is not supported. The second object is supported and the property is writable. The objective is to verify that an appropriate error response is returned and that all writes after the first failed write attempt do not take place.

Configuration Requirements: If the IUT supports any writable scalar properties that are not commandable it shall be configured with one for use in this test. If no such properties are supported the IUT shall be configured with a writable array or commandable property and the test steps modified to account for this variation. In the test description O2 and P2 will be used to designate the writable object and property having value X used for this test. The designation Bad Object will be used to indicate an object that is not supported or not present in IUT databaseP1 is any valid Property Identifier.

Test Steps:
1. VERIFY (O2), P2 = X
2. TRANSMIT WritePropertyMultiple-Request,
   'Object Identifier' = BadObject,
   'Property Identifier' = P1,
   'Property Value' = (any valid value of the appropriate datatype for this property subject to the restrictions specified in the EPICS as defined in 4.4.2)
   'Object Identifier' = O2,
   'Property Identifier' = P2,
   'Property Value' = (any valid value not equal to X),
3. RECEIVE WritePropertyMultiple-Error,
   'Error Class' = OBJECT,
   'Error Code' = UNKNOWN_OBJECT,
   'Object Identifier' = BadObject,
   'Property Identifier' = P1 |
   (RECEIVE WritePropertyMultiple-Error,
   'Error Class' = OBJECT,
   'Error Code' = UNSUPPORTED_OBJECT_TYPE,
   'Object Identifier' = BadObject,
   'Property Identifier' = P1)
4. VERIFY (O2), P2 = X

9.23.2.X4 Writing first element of 'List of Write Access Specifications' with a Write Access Error

Purpose: To verify the ability to correctly execute a WritePropertyMultiple service request for which the first element of the 'List of Write Access Specifications' contains a specification for a read only property and all writes after the first failed write attempt do not take place.

Test Concept: An attempt is made to write to two properties in a single object. The first property is supported but read only. The second property is supported and writable. The objective is to verify that an appropriate error response is returned and that all writes after the first failed write attempt do not take place.

Configuration Requirements: If the IUT supports any writable scalar properties that are not commandable it shall be configured with one for use in this test. If no such properties are supported the IUT shall be configured with a writable array or commandable property and the test steps modified to account for this variation. In the test description O1 will be used to designate the object, P1 the read only property having value X, P2 the writable property having value Y used for this test.

Test Steps:

```
1.  VERIFY (O1), P1= X
2.  VERIFY (O2), P2=Y
3.  TRANSMIT WritePropertyMultiple-Request,
        'Object Identifier' = O1,
        'Property Identifier' = P1,
        'Property Value' = X ),
        'Property Identifier' = P2,
        'Property Value' = (any valid value not equal to Y)
4.  RECEIVE WritePropertyMultiple-Error,
        'Error Class' = PROPERTY,
        'Error Code' = WRITE_ACCESS_DENIED,
        'Object Identifier' = O1,
        'Property Identifier' = P1
5.  VERIFY (O1), P2 = Y
```

9.23.2.X5 WritePropertyMultiple Reject Test for first element of 'List of Write Access Specifications'

Purpose: This test case verifies that if IUT does sends a Reject-PDU or Error-PDU then the write attempt for the remaining element of '**List of Write Access Specifications'** do not take place.

Test Concept: Two writable properties, P1 having value X and P2 having value Y are written to the IUT but the portion of the WritePropertyMultiple specifying P1 is made invalid by omitting the 'Property Value' parameter. The value of the properties are checked to ensure that it has not changed.

Test Steps:

```
1.  VERIFY (O1), P1= X
2.  VERIFY (O2), P2=Y
3.  TRANSMIT WritePropertyMultiple-Request,
        'Object Identifier' = O1,
        'Property Identifier' = P1,
        'Object Identifier' = O2,
        'Property Identifier' = P2
        'Property Value' = (Any valid value not equal to Y))
4.  RECEIVE WritePropertyMultiple-Error,
        'Error Class' = SERVICES,
        'Error Code' = INVALID_TAG
        'Object Identifier' = O1
        'Property Identifier' = P1) |
    (RECEIVE BACnet-Reject-PDU,
        'Reject Reason' = INVALID_TAG | MISSING_REQUIRED_PARAMETER |
                          INCONSISTENT_PARAMETERS | INVALID_PARAMETER_DATA_TYPE |
                          TOO_MANY_ARGUMENTS)
4.  VERIFY (O1), P1 = X
5.  VERIFY (O2), P2 = Y
```

BTL-15.2k-6: Add Testing for Read-Only Recipient_List Property [Interim Tests]

Overview:

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

Changes:

<Changes to Notification Class Object in the Checklist and Test Plan were removed after review because they are already included in Test Package 15.2>

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

5.2 Alarm and Event - Notification - Internal - B

5.2.1 Base Requirements

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

BTL - 7.3.1.10 - Event_Enable Tests		
	Test 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.
BTL - 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 also tested on Event Enrollment objects.
	Testing Hints	
135.1-2009 - 8.4 - ConfirmedEventNotification Service Initiation Tests		
	Test Conditionality	Must be executed unless IUT contains only read-only Recipient_List properties and does not claim Notification Forwarder objects. Any of the 8.4 tests can be used to ensure that the IUT properly generates ConfirmedEventNotification requests. The specific tests that can be executed are detailed under the test cases for the specific algorithms. As long as one of the tests is executed using ConfirmedEventNotifications, then this test case shall be satisfied.
	Test Directives	
	Testing Hints	
135.1-2009 - 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	

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

7.3.2.21.3.1 ValidDays Test

…

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

7.3.2.21.3.2 FromTime and ToTime Test

…

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

7.3.2.21.3.3 IssueConfirmedNotifications Test

…

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

7.3.2.21.3.4 Transitions Test

…

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

7.3.2.21.3.5 Recipient_List Property Supports Device Identifier Recipients Test

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

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

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

Test Steps:

1. WRITE N.RecipientList = ({all days, all times, D, any process ID, FALSE, all transitions})
2. MAKE (the event generating object, E, transition)
3. BEFORE Notification Fail Time plus the amount of time the IUT takes to perform device discovery

```
        RECEIVE UnconfirmedEventNotification-Request,
            'Process Identifier' =             (the valid process ID from step 1),
            'Initiating Device Identifier' =   IUT,
            'Event Object Identifier' =        E,
            'Time Stamp' =                     (any valid time stamp),
            'Notification Class' =             (N's instance),
            'Priority' =                       (any valid priority),
            'Event Type' =                     (any valid event type),
            'Notify Type' =                    ALARM | EVENT,
            'AckRequired' =                    TRUE | FALSE,
            'From State' =                     (any valid event state),
            'To State' =                       (any valid event state),
            'Event Values' =                   (values appropriate to the event type)
```

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

7.3.2.21.3.6 Recipient_List Property Supports Network Address Recipients

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

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

Test Steps:

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

[Add new test into BTL Specified Tests.]

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

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

Test Concept: Read the Recipient_List of the Notification Class objects and check that the length is 1, the Recipient is local broadcast, Valid Days are all days, From Time and To Time are the entire day, Process Identifier is 0, Issue Confirmed

Notification is False and Transitions is set to all transitions. This test is only applied to IUT devices that have read-only Notification Class object Recipient_List properties, and which do not contain internal Notification Forwarder objects.

Test Steps:

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

BTL-15.2k-7: Add Testing for RPM on Array Properties [BTLWG-68]

Overview:
Reason for Change: The existing testcases of ReadPropertyMultiple contain no testing of BACnetARRAY properties, except their size at index 0. One new test that is repeated for all supported BACnetARRAY properties, is added. The new testcase is generalized and not only applied to Object_List property (as 135.1-2013 test 9.20.1.10 had been).

Changes:

[In Test Plan sections 4.4.1 add a test case as shown]

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

BTL - 9.20.1.X2 - ReadPropertyMultiple Array Properties		
	Test Conditionality	Must be executed.
	Test Directives	Repeat for all supported BACnetARRAY properties
	Testing Hints	

[In BTL Specified Tests in section 9.20.1 add a test case as shown]

9.20.1.X2 ReadPropertyMultiple Array Properties

Purpose: To verify that the IUT can execute ReadPropertyMultiple service requests when the requested property is an array, when its size as well as when a single element of the array is requested. Another request is made to read an element of an array where the array index is out of range.

Configuration Requirement: O1 is any object in the IUT database having array property P1 having size X.

Test Steps:

```
1.  VERIFY P1 = X, ARRAY INDEX = 0
2.  IF (X>0) THEN
3.  TRANSMIT ReadPropertyMultiple-Request,
        'Object Identifier' = O1,
        'Property Identifier' = P1,
        'Property Array Index' = 1
4.  RECEIVE ReadPropertyMultiple-ACK,
        'Object Identifier' = O1,
        'Property Identifier' = P1,
        'Property Array Index' = 1,
        'Property Value' = (V, any valid value of the correct data type for property P1)
5.  TRANSMIT ReadPropertyMultiple-Request,
        'Object Identifier' = O1,
        'Property Identifier' = P1,
        'Property Array Index' = X,
6.  RECEIVE ReadPropertyMultiple-ACK,
        'Object Identifier' = O1,
        'Property Identifier' = P1,
        'Property Array Index' = X,
        'Property Value' = (V, any valid value of the correct data type for property P1)
7.  CHECK (V is any valid value of the correct data type for property P1)
8.  TRANSMIT ReadPropertyMultiple-Request,
        'Object Identifier' = O1,
        'Property Identifier' = P1,
        'Property Array Index' = (X+1)
9.  RECEIVE ReadPropertyMultiple-Error,
        Error Class = PROPERTY,
        Error Code = INVALID_ARRAY_INDEX
```

BTL-15.2k-8: Add Testing for Pattern and Non-Pattern Properties with WPM [BTLWG-159]

Overview:

Reason for Change: The existing test plan have test cases for pattern and non-pattern using WriteProperty service. Added the WPM version for each non-Pattern and non-Pattern DateRange test.

Changes:

[In BTL Test Plan, add the test cases as specified below]

3.8 Calendar Object

3.8.1 Base Requirements

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

…		
BTL - 9.23.2.X12 - BACnetDateRange Non-Pattern Properties Test using WritePropertyMultiple service		
	Test Conditionality	This test shall only be applied to devices claiming Protocol_Revision 11 or higher and which supports execution of WritePropertyMultiple.
	Test Directives	Apply to the Date_List property.
	Testing Hints	

3.10 Device Object

3.10.5 Contains a Writable Local_Date Property

The IUT contains, or can be made to contain, a Device object that contains a writable Local_Date property.

…		
BTL - 9.23.2.X9 - Date Non-Pattern Properties Test using WritePropertyMultiple service		
	Test Conditionality	This test shall only be applied to devices claiming Protocol_Revision 11 or higher and which supports execution of WritePropertyMultiple.
	Test Directives	Apply to the Local_Date property.
	Testing Hints	

3.10.6 Contains a Writable Local_Time Property

The IUT contains, or can be made to contain, a Device object that contains a writable Local_Time property.

…		
BTL - 9.23.2.X10 - Time Non-Pattern Properties Test using WritePropertyMultiple service		
	Test Conditionality	This test shall only be applied to devices claiming Protocol_Revision 11 or higher and which supports execution of WritePropertyMultiple.
	Test Directives	Apply to the Local_Time property in Device object
	Testing Hints	

3.17 Notification Class Object

3.17.3 Supports Writable Recipient_List Properties

The IUT supports Recipient_List properties that are modifiable via write services. All required values must be accepted in write requests.

…		
BTL - 9.23.2.X10 - Time Non-Pattern Properties Test using WritePropertyMultiple service		
	Test Conditionality	This test shall only be applied to devices claiming Protocol_Revision 11 or higher and which supports execution of WritePropertyMultiple.
	Test Directives	Apply to the fromTime and again to the toTime element in a Recipient_List property in a Notification Class.
	Testing Hints	

3.27 Date Value Object

3.27.4 Contains a Writable Present_Value Property

The IUT contains, or can be made to contain, a Date Value object that contains a writable Present_Value property.

…		
BTL – 7.3.X4 - Date Non-Pattern Properties Test using WritePropertyMultiple service		
	Test Conditionality	This test shall only be applied to devices claiming Protocol_Revision 11 or higher and which supports execution of WritePropertyMultiple.
	Test Directives	Apply to the Present_Value property in a Date Value object.
	Testing Hints	

3.27.5 Contains a Writable Relinquish_Default Property

The IUT contains, or can be made to contain, a Date Value object that contains a writable Relinquish_Default property.

…		
BTL - 9.23.2.X9 - Date Non-Pattern Properties Test using WritePropertyMultiple service		
	Test Conditionality	This test shall only be applied to devices claiming Protocol_Revision 11 or higher and which supports execution of WritePropertyMultiple.
	Test Directives	Apply to the Relinquish_Default property in a Date Value object.
	Testing Hints	

3.29 Date Time Value Object

3.29.4 Contains a Writable Present_Value Property

The IUT contains, or can be made to contain, a DateTime Value object that contains a writable Present_Value property.

…		
BTL - 9.23.2.X11 - DateTime Non-Pattern Properties Test using WritePropertyMultiple service		
	Test Conditionality	This test shall only be applied to devices claiming Protocol_Revision 11 or higher and which supports execution of WritePropertyMultiple.
	Test Directives	Apply to the Present_Value property in a DateTime Value object.
	Testing Hints	

3.29.5 Contains a Writable Relinquish_Default Property

The IUT contains, or can be made to contain, a DateTime Value object that contains a writable Relinquish_Default property.

…		
BTL - 9.23.2.X11 - DateTime Non-Pattern Properties Test using WritePropertyMultiple service		
	Test Conditionality	This test shall only be applied to devices claiming Protocol_Revision 11 or higher and which supports execution of WritePropertyMultiple.
	Test Directives	Apply to the Relinquish_Default property in a DateTime Value object.
	Testing Hints	

3.35.4 Contains a Writable Present_Value Property

The IUT contains, or can be made to contain, a Time Value object that contains a writable Present_Value property.

…		
BTL - 9.23.2.X10 - Time Non-Pattern Properties Test using WritePropertyMultiple service		
	Test Conditionality	This test shall only be applied to devices claiming Protocol_Revision 11 or higher and which supports execution of WritePropertyMultiple.
	Test Directives	Apply to the Present_Value property in a Time Value object.
	Testing Hints	

3.35.5 Contains a Writable Relinquish_Default Property

The IUT contains, or can be made to contain, a Time Value object that contains a writable Relinquish_Default property.

…		
BTL - 9.23.2.X10 - Time Non-Pattern Properties Test using WritePropertyMultiple service		
	Test Conditionality	This test shall only be applied to devices claiming Protocol_Revision 11 or higher and which supports execution of WritePropertyMultiple.

	Test Directives	Apply to the Relinquish_Default property in a Time Value object.
	Testing Hints	

5.14 Alarm and Event Management - Event Log - Internal - B

5.14.5 Supports Start_Time and Stop_Time Properties

The IUT can be made to start and stop logging using these properties.

These properties are required to be present and writable in Event Log objects, if either is present.

…		
BTL - 9.23.2.X11 - DateTime Non-Pattern Properties Test using WritePropertyMultiple service		
	Test Conditionality	This test shall only be applied to devices claiming Protocol_Revision 11 or higher and which supports execution of WritePropertyMultiple.
	Test Directives	Apply to the Start_Time and again to the Stop_Time properties in an Event Log object.
	Testing Hints	

6.4 Scheduling - Internal - B

6.4.1 Base Requirements

Base requirements must be met by any IUT claiming conformance to this BIBB. (The BIBB requires, among other things, support for either TimeSynchronization-Request or UTCTimeSynchronization-Request execution; these are tested by the Device Management tests.)

…		
BTL - 9.23.2.X10 - Time Non-Pattern Properties Test using WritePropertyMultiple service		
	Test Conditionality	This test shall only be applied to devices claiming Protocol_Revision 11 or higher and which supports execution of WritePropertyMultiple.
	Test Directives	Apply to the time portion of BACnetTimeValues in the Exception_Schedule property, then apply to the time portion of BACnetTimeValues in the Weekly_Schedule property.
	Testing Hints	
BTL - 9.23.2.X12 - BACnetDateRange Non-Pattern Properties Test using WritePropertyMultiple service		
	Test Conditionality	This test shall only be applied to devices claiming Protocol_Revision 11 or higher and which supports execution of WritePropertyMultiple.
	Test Directives	Apply to BACnetCalendarEntry in the Exception_Schedule property in the BACnetDateRange form.
	Testing Hints	

6.4.4 Supports Configurable Effective_Period

The IUT supports the Effective_Period property and it is configurable.

…		
BTL - 9.23.2.X12 - BACnetDateRange Non-Pattern Properties Test using WritePropertyMultiple service		
	Test Conditionality	This test shall only be applied to devices claiming Protocol_Revision 11 or higher and which supports execution of WritePropertyMultiple.
	Test Directives	Apply to the Effective_Period property.
	Testing Hints	

6.6 Scheduling - Weekly Schedule - Internal - B

6.6.7 Supports Configurable Effective_Period

The IUT supports the Effective_Period property and it is configurable.

…		
BTL - 9.23.2.X12 - BACnetDateRange Non-Pattern Properties Test using WritePropertyMultiple service		
	Test Conditionality	This test shall only be applied to devices claiming Protocol_Revision 11 or higher and which supports execution of WritePropertyMultiple.

	Test Directives	Apply to the Effective_Period property.
	Testing Hints	

7.3. Trending - View and Modify Trends - I - B

7.3.7 Supports Start_Time and Stop_Time Properties

The IUT can be made to start and stop logging using these properties.

These properties are required to be present and writable in trend log objects that are trending a BACnet property.

…		
BTL - 9.23.2.X11 - DateTime Non-Pattern Properties Test using WritePropertyMultiple service		
	Test Conditionality	This test shall only be applied to devices claiming Protocol_Revision 11 or higher and which supports execution of WritePropertyMultiple.
	Test Directives	Apply to the Start_Time and again to the Stop_Time properties in a Trend Log object.
	Testing Hints	

7.7 Trending - View and Modify Multiple Values - I - B

7.7.7 Supports Start_Time and Stop_Time Properties

The IUT can be made to start and stop logging using these properties.

If present these properties are required to be writable.

…		
BTL - 9.23.2.X11 - DateTime Non-Pattern Properties Test using WritePropertyMultiple service		
	Test Conditionality	This test shall only be applied to devices claiming Protocol_Revision 11 or higher and which supports execution of WritePropertyMultiple.
	Test Directives	Apply to the Start_Time and again to the Stop_Time properties in a Trend Log object.
	Testing Hints	

[Addition in Test Plan for new test case]

9.23.2.X9 Date Non-Pattern Properties Test using WritePropertyMultiple service

Purpose: To verify that the property being tested does not accept special date field values.

Test Concept: The property being tested, P1, is written with each of the special date field values to ensure that the property does not accept them. A date is selected which is within the date range that the IUT will accept for the property. The value, V1, written to the property is the date D1 with one of its fields replaced with one of the date special values. If the property is a complex datatype, the other fields in the value shall be set within the range accepted by the IUT. This test shall only be applied to devices claiming Protocol_Revision 11 or higher.

Test Steps:

```
1.  REPEAT SV = (year unspecified, month unspecified, day of month unspecified,
            day of week unspecified, odd months, even months, last day of month,
            even days, odd days) DO {
2.      TRANSMIT WritePropertyMultiple-Request
            'Object Identifier' = O1,
            'Property Identifier' = P1,
            'Property Value' = (V1 updated with the special value SV)
3.      RECEIVE WritePropertyMultiple-Error,
            'Error Class' =             PROPERTY,
            'Error Code' =              VALUE_OUT_OF_RANGE,
            'Object Identifier' =   Object1,
            'Property Identifier' = P1)
        | (BACnet-Reject-PDU
            'Reject Reason' = INVALID_PARAMETER_DATATYPE)
        | (BACnet-Reject-PDU
            'Reject Reason'= INVALID_TAG)
 }
```

Notes to Tester: if P1 is an array, then a non-zero array index may be provided in the TRANSMIT and the same array index observed in the WritePropertyMultiple-Error.

9.23.2.X10 Time Non-Pattern Properties Test using WritePropertyMultiple service
Purpose: To verify that the property being tested does not accept special time field values.

Test Concept: The property being tested, P1, is written with each of the special time field values to ensure that the property does not accept them. A time is selected which is within the time range that the IUT will accept for the property. The value, V1, written to the property is the time T1 with one of its fields replaced with one of the time special values. If the property is a complex datatype, the other fields in the value shall be set within the range accepted by the IUT. This test shall only be applied to devices claiming Protocol_Revision 11 or higher.

Test Steps:

```
1.  REPEAT SV = (hour unspecified, minute unspecified, second unspecified, hundredths unspecified) Do {
2.      TRANSMIT WritePropertyMultiple-Request
            'Object Identifier' = O1,
            'Property Identifier' = P1,
            'Property Value' = (V1 updated with the special value SV)
3.      RECEIVE WritePropertyMultiple-Error,
            'Error Class' =             PROPERTY,
            'Error Code' =              VALUE_OUT_OF_RANGE,
            'Object Identifier' =   Object1,
            'Property Identifier' = P1
        | (BACnet-Reject-PDU
            'Reject Reason' = INVALID_PARAMETER_DATATYPE)
        | (BACnet-Reject-PDU
            'Reject Reason'= INVALID_TAG)
}
```

Notes to Tester: if P1 is an array, then a non-zero array index may be provided in the TRANSMIT and the same array index observed in the WritePropertyMultiple-Error.

9.23.2.X11 DateTime Non-Pattern Properties Test using WritePropertyMultiple service

Purpose: To verify that the property being tested does not accept special date field values.

Test Concept: The property being tested, P_1, is written with each of the special datetime field values to ensure that the property does not accept them. A datetime DT_1 is selected which is within the range that the IUT will accept for the property. The value, V_1, written to the property is the datetime DT_1 with one of its fields replaced with one of the date or time special values. If the property is a complex datatype, the other fields in the value shall be set within the range accepted by the IUT. This test shall only be applied to devices claiming Protocol_Revision 11 or higher.

Test Steps:

```
1.  REPEAT SV = (year unspecified, month unspecified, day of month unspecified, odd months, even months, last day of
    month, even days, odd days, hour unspecified, minute unspecified, second unspecified, hundredths unspecified) DO {
2.      TRANSMIT WritePropertyMultiple-Request,
            'Object Identifier' = O1,
            'Property Identifier' = P1,
            'Property Value' = (DT1 updated with the special value SV)
3.      RECEIVE WritePropertyMultiple-Error,
            'Error Class' =             PROPERTY,
            'Error Code' =              VALUE_OUT_OF_RANGE,
            'Object Identifier' =   Object1,
            'Property Identifier' = P1)
        | (BACnet-Reject-PDU
            'Reject Reason' = INVALID_PARAMETER_DATATYPE)
        | (BACnet-Reject-PDU
            'Reject Reason'= INVALID_TAG)
    }
```

Notes to Tester: if P1 is an array, then a non-zero array index may be provided in the TRANSMIT and the same array index observed in the WritePropertyMultiple-Error.

9.23.2.X12 BACnetDateRange Non-Pattern Properties Test using WritePropertyMultiple service

Purpose: To verify that the property being tested does not accept special date field values, except for fully unspecified start of the range or fully unspecified end of the range, or both.

Test Concept: A BACnetDateRange property, or property that is a complex datatype containing BACnetDateRange P1 is written with each of the special field values to ensure that the property does not accept them. Each half of the dateRange DR1 is selected so it is within the range that the IUT will accept for the property. The value, V1 written to the property is the dateRange DR1 with one of its fields replaced with one of the date special values. If the property is a complex datatype, the other fields in the value shall be set within the range accepted by the IUT. This test shall only be applied to devices claiming Protocol_Revision 11 or higher.

Test Steps:

```
1.  REPEAT SV = (year unspecified, month unspecified, day of month unspecified,
            day of week unspecified, odd months, even months, last day of month,
            even days, odd days) DO {
2.      TRANSMIT WritePropertyMultiple-Request,
            'Object Identifier' = O1,
            'Property Identifier' = P1,
            'Property Value' = (DR1 with startDate updated with special value SV)
3.      RECEIVE WritePropertyMultiple-Error,
            'Error Class' =          PROPERTY,
            'Error Code' =           VALUE_OUT_OF_RANGE,
            'Object Identifier' =    Object1,
            'Property Identifier' =  P1
        | (BACnet-Reject-PDU
            'Reject Reason' = INVALID_PARAMETER_DATATYPE)
        | (BACnet-Reject-PDU
            'Reject Reason'= INVALID_TAG)
4.      TRANSMIT WritePropertyMultiple-Request,
            'Object Identifier' = O1,
            'Property Identifier' = P1,
            'Property Value' = (DR1 with endDate updated with special value SV)
5.      RECEIVE WritePropertyMultiple-Error,
            'Error Class' =          PROPERTY,
            'Error Code' =           VALUE_OUT_OF_RANGE,
            'Object Identifier' =    Object1,
            'Property Identifier' =  P1)
        | (BACnet-Reject-PDU
            'Reject Reason' = INVALID_PARAMETER_DATATYPE)
        | (BACnet-Reject-PDU
            'Reject Reason'= INVALID_TAG)
}
```

Notes to Tester: if P1 is an array, then a non-zero array index may be provided in the TRANSMIT and the same array index observed in the WritePropertyMultiple-Error.

BTL-15.2k-9: Add Clock-Aligned Testing [BTLWG-429]

Overview:

Clock-aligned interval logging and Interval_Offset during logging have had no existing tests.
With clarity by IC 135-2008-21 and IC 135-2010-3, the testing for proper behavior of Align_Intervals and Interval_Offset can be added to the BTL Test Package.

Typographic convention: These wholly new tests are shown without highlight ~~strike-through~~, or *italic.*

Changes:

[In BTL Specified Tests, add two new tests]

7.3.2.24.X8 Clock-Aligned Logging

Purpose: To verify that logged trend records have timestamps aligned to that interval, when Align_Intervals is TRUE and Log_Interval is a factor of (divides without remainder) a day.

Test Concept: For this test, select two evenly divisible factors. Write each to Log_Interval in the test. Trend records are logged, and checked that those are aligned to the Log_Interval. This is done twice to ensure that different interval frequency behavior is verified. This test does not employ Log_Interval values which are not one of the evenly divisible factors.

Configuration Requirements: Start_Time, if present, shall be configured with a date and time preceding the beginning of the test. Stop_Time, if present shall be configured in order that it occurs after the end of the test. Stop_When_Full, if configurable, shall be set to FALSE. Enable is initially FALSE. Interval_Offset is set to zero. Align_Intervals is set to TRUE. Triggering of non periodic log records should not occur during this test. Logging_Mode is POLLED. X1 and X2 are each a value which the IUT supports for which the standard mandates the alignment behavior.

Test Steps:

1. VERIFY (Log_Buffer contains 1 entry, and it is the buffer-purged event)
2. WRITE Log_Interval = X1
3. WRITE Enable = TRUE
4. MAKE (logging object collect at least 2 records)
5. WRITE Enable = FALSE
6. VERIFY (Log_Buffer contains at least 5 entries, and at least two data records)
7. CHECK (that the timestamp of each data record, since the Log_Interval was written, is a multiple of X1)
8. WRITE Log_Interval = (X2, any value which requires alignment behavior, that was not already chosen)
9. WRITE Enable = TRUE
10. MAKE (logging object collect at least 2 more records)
11. WRITE Enable = FALSE
12. VERIFY (Log_Buffer has collected two or more additional data records and two or more log-status records)
13. CHECK (that the timestamp are multiples of X2 for all data records collected, since the write with X2)

Notes to Tester: The values for Log_Interval which require alignment are those for which the standard mandates the alignment behavior, where 8,640,000 modulo Log_Interval is zero.

7.3.2.24.X9 Logging Interval_Offset

Purpose: To verify that timestamps abide by the Interval_Offset.

Test Concept: Log_Interval is set to a value which the IUT supports which is a factor of (divides without remainder) a day and which is greater than 3 seconds.

Interval_Offset is first set to a non-zero value less than Log_Interval. After logging some records, their timestamps are checked. The logging is stopped. Interval_Offset is set to a value which the IUT supports greater than Log_Interval, logging is re-enabled, and the timestamps again are checked.

Configuration Requirements: Align_Intervals is set to TRUE. The Log_DeviceObjectProperty property in a Trend Log or in a Trend Log Multiple, is configured to the property or properties monitored. Start_Time, if present, shall be configured with a date and time preceding the beginning of the test. Stop_Time, if present shall be configured in order that it occur after the end of the test. Stop_When_Full, if configurable, shall be set to FALSE. Enable is initially FALSE. Align_Intervals is set to TRUE.

Triggering of non periodic log records should not occur during this test. If the Interval_Offset cannot be set to a value which the IUT supports greater than Log_Interval, then steps 11 through the end of this test are skipped. Logging_Mode is POLLED. An evenly divisible value is a value for which the standard mandates the alignment behavior.

Test Steps:

1. WRITE Record_Count = 0
2. VERIFY (Log_Buffer contains 1 entry, and it is the buffer-purged event)
3. WRITE Log_Interval = (any evenly divisible value greater than 3 seconds)
4. WRITE Interval_Offset = (any value, between 2 seconds and Log_Interval - 1 seconds)
5. WRITE Enable = TRUE
6. MAKE (logging object collect at least 2 records)
7. WRITE Enable = FALSE
8. VERIFY (Log_Buffer contains two or more data records and at least three log-status)
9. CHECK (the timestamp for the data records have a fixed offset, determined by Log_Interval and Interval_Offset)
10. WRITE Interval_Offset = (any value greater than Log_Interval)
11. WRITE Enable = TRUE
12. MAKE (logging object collect at least 2 records)
13. WRITE Enable = FALSE
14. VERIFY (Log_Buffer has collected two or more additional data records and two or more log-status entries)
15. CHECK (the timestamp for data records collected since the Interval_Offset was last written, have Log_Interval between records, at a fixed offset of Interval_Offset modulo Log_Interval)

Note to tester: Interval_Offset in logging objects, and Log_Interval are each an Unsigned number of hundredths of seconds. Excellent choices are 400, 500, 600, 1000, or 1200. When Interval_Offset is larger than Log_Interval, then Interval_Offset modulo Log_Interval, is smaller than Log_Interval.

[Revise the references in BTL Test Plan to these two tests being added in BTL Specified Tests, in two different BIBBs, T-VMT-I-B, and T-VMMV-I-B, as shown]

7.3.8 Supports Clock-aligned Logging

The IUT can be made to gather trend data with clock-aligned Timestamps.

BTL - 7.3.2.24.X8 - Clock-aligned logging		
	Test Conditionality	Must be executed.~~This test is not yet defined and shall be skipped.~~
	Test Directives	
	Testing Hints	
BTL - 7.3.2.24.X9 - Logging Interval_Offset		
	Test Conditionality	Must be executed.~~This test is not yet defined and shall be skipped.~~
	Test Directives	
	Testing Hints	

7.7.6 Supports Clock-aligned Logging

The IUT can be made to gather trend data with clock-aligned Timestamps.

BTL - 7.3.2.24.X8 - Clock-aligned logging		
	Test Conditionality	Must be executed.~~This test is not yet defined and shall be skipped.~~
	Test Directives	
	Testing Hints	
BTL - 7.3.2.24.X9 - Logging Interval_Offset		
	Test Conditionality	Must be executed.~~This test is not yet defined and shall be skipped.~~
	Test Directives	
	Testing Hints	