



# **BACnet<sup>®</sup> TESTING LABORATORIES ADDENDA**

## **Addendum fix1 to BTL Test Package 16.1**

**Revision 6  
Revised 6/18/2020**

Approved by the BTL Working Group on May 21, 2020.  
Approved by the BTL Working Group Voting Members on July 22, 2020.  
Published on September 2, 2020.

**[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.

BTL-16.1fix1-1: Router Discovery Timeout Fixes - BTLWG-349 [CR-0420].....	2
BTL-16.1fix1-2: Fix for Event_Algorithm_Inhibit_Ref Tests - BTLWG-599 [CR-0428] .....	4
BTL-16.1fix1-3: Clarify How to Read Large Properties - BTLWG-601 [CR-0413] .....	8
BTL-16.1fix1-4: Fix Out_Of_Service Accumulator Test - BTLWG-610 [CR-0463].....	9
BTL-16.1fix1-5: Fix Out_Of_Service Usage in Value Objects - BTLWG-737 [CR-0456].....	10
BTL-16.1fix1-6: Fix Reliability_Inhibit Tests - BTLWG-736 & BTLWG-741 [CR-0455] .....	13
BTL-16.1fix1-7: Fix Adjust_Value Write Test - BTLWG-751 [CR-0459] .....	15
BTL-16.1fix1-8: Correct Test Plan Entry for AE-N-E-B - BTLWG-880 .....	16
BTL-16.1fix1-9: COV Subscription Test Changes - BTLWG-841 [CR-0458].....	17
BTL-16.1fix1-10: Update Conditionality for Event_Detection_Enable Test - BTLWG-653 [CR-0448] .....	19
BTL-16.1fix1-11: Improve Change of Value Notification Test Names - BTLWG-732.....	20

In the following document, language to be added to existing clauses within the BTL Test Package 16.1 is indicated through the use of *italics*, while deletions are indicated by ~~strike through~~. Where entirely new subclauses are proposed to be added, plain type is used throughout

In contrast, 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-16.1fix1-1: Router Discovery Timeout Fixes - BTLWG-349 [CR-0420]**

### **Overview:**

In BTLWG CR-0420 it was decided that tests 135.1-2013 10.2.3.6.1 and 10.2.3.6.2 were correct but should be improved to mention the timeout for router discovery. Reference: SSPC Interpretation Request 135-2016-13.

### **Changes:**

[In BTL Specified Tests, add tests 10.2.3.6.1 and 10.2.3.6.2 from 135.1-2013 and correct them as shown.]

#### **10.2.3.6.1 Failed Attempt to Locate Router**

Purpose: To verify that the IUT will attempt to locate a router to an unknown network. Upon failing to locate such a router the IUT will transmit a Reject-Message-To-Network to the source device.

*Configuration Requirements: The IUT shall be configured to know only about its directly connected networks.*

*TOMax: vendor defined time the router waits before sending a Reject-Message-to-Network request.*

*TOMin: vendor defined minimum time the router waits for a response to the Who-Is-Router-To-Network request.*

*Notes to Tester: The standard does not provide any guidance on how long a router should wait before declaring that the attempt to locate the next router failed. While there is no explicit minimum time, it is expected that routers wait long enough that the attempt would succeed if the next hop router responded immediately.*

#### **Test Steps:**

1. TRANSMIT PORT A, DA = IUT,  
SA = R1-5, DNET = 3,  
DADR = D3D,  
SNET = 5,  
SADR = D5F,  
Hop Count = 254,  
BACnet-Confirmed-Request-PDU,  
'Service Choice' = ReadProperty-Request,  
'Object Identifier' = (any object identifier),  
'Property Identifier' = (any property of the specified object)
2. RECEIVE PORT B,  
DESTINATION = LOCAL BROADCAST, SOURCE = IUT, Who-Is-Router-To-Network,  
Network Number = 3
3. *WAIT TOMin*
4. *BEFORE TOMax*  
RECEIVE PORT A, DA = R1-5,  
SOURCE = IUT,  
DNET = 5,  
DADR = D5F,  
Hop Count = 255,  
Reject-Message-To-Network,  
Reject Reason = 1 (unknown destination network), DNET = 3

#### **10.2.3.6.2 Successful Attempt to Locate Router**

Purpose: To verify that the IUT will attempt to locate a router to an unknown network. When successful it forwards the message to the next router on the path.

*Configuration Requirements: The IUT shall be configured to know only about the directly-connected networks.*

*TOMin: vendor defined minimum time the router waits for a response to the Who-Is-Router-To-Network request.*

*Notes to Tester: The standard does not provide any guidance on how long a router should wait before declaring that the attempt to locate the next router failed. While there is no explicit minimum time, it is expected that routers wait long enough that the attempt would succeed if the next hop router responded immediately.*

Test Steps:

1. TRANSMIT PORT A, DA = IUT,  
SA = R1-5,  
DNET = 3,  
DADR = D3D,  
SNET = 5,  
SADR = D5F,  
Hop Count = 254,  
BACnet-Confirmed-Request-PDU,  
'Service Choice' = ReadProperty-Request,  
'Object Identifier' = (any object identifier),  
'Property Identifier' = (any property of the specified object)
2. RECEIVE PORT B,  
DESTINATION = LOCAL BROADCAST, SOURCE = IUT, Who-Is-Router-To-Network,  
Network Number = 3
3. *WAIT any time less than T<sub>Omin</sub>*
4. TRANSMIT PORT B,  
DESTINATION = LOCAL BROADCAST, SOURCE = R2-3, I-Am-Router-To-Network,  
Network Numbers = 3
5. RECEIVE PORT B, SA = R2-3,  
SA = IUT, DNET = 3, DADR = D3D,  
SNET = 5,  
SADR = D5F,  
Hop Count = (any integer x:  $0 < x < 254$ ), BACnet-Confirmed-Request-PDU,  
'Service Choice' = ReadProperty-Request,  
'Object Identifier' = (the object identifier used in step 1), 'Property Identifier' = (the property identifier used in step 1)

**BTL-16.1fix1-2: Fix for Event\_Algorithm\_Inhibit\_Ref Tests - BTLWG-599 [CR-0428]****Overview:**

CR-0428 response specified that in test 7.3.1.X7.2 to add to Configuration Requirements, mentioning that if both the Event\_Algorithm\_Inhibit and Event\_Algorithm\_Inhibit\_Ref optional properties are omitted, that is another conditional situation in which the test shall be skipped.

**Changes:**

[ In BTL Specified Tests, in test 7.3.1.X7.1 [7.3.1.20.1] and 7.3.1.X7.2 [7.3.1.20.2], derive from the versions in 135.1-2013q [135.1-2019] to reference the correct test, and to correct the Configuration Requirements (requires presence, rather than requiring absence). ]

**7.3.1.X7 [7.3.1.20] Event\_Algorithm\_Inhibit\_Ref Tests****7.3.1.X7.1 [7.3.1.20.1] Event\_Algorithm\_Inhibit\_Ref Test**

Reason for Change: The version in 135.1-2013q has incorrect test reference, and incorrect Configuration Requirements.

Purpose: To verify that the object referenced by Event\_Algorithm\_Inhibit\_Ref controls Event\_Algorithm\_Inhibit and thus whether or not the event state detection algorithm is executed.

Test Concept: Execute test 7.3.1.X62.1 [7.3.1.19.1] against an object O2 which supports both Event\_Algorithm\_Inhibit\_Ref and Event\_Algorithm\_Inhibit and instead of writing Event\_Algorithm\_Inhibit, write the property referenced by Event\_Algorithm\_Inhibit\_Ref to change the value in the Event\_Algorithm\_Inhibit property.

Configuration Requirements: If the IUT has no object *which has an* ~~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.

**7.3.1.X7.2 [7.3.1.20.2] Event\_Algorithm\_Inhibit Writable Test**

Reason for Change: The version in 135.1-2013q didn't express the correct Configuration Requirements.

Purpose: To verify that *whenever* ~~if~~ the Event\_Algorithm\_Inhibit\_Ref property is absent or is uninitialized then the Event\_Algorithm\_Inhibit property shall be writable.

Configuration Requirements: Select an event-initiating object, O1 *which has an Event\_Algorithm\_Inhibit property, but in* which Event\_Algorithm\_Inhibit\_Ref property is absent or is uninitialized. If the IUT has no such object, this test shall be skipped.

**Test Steps:**

1. WRITE Event\_Algorithm\_Inhibit = TRUE
2. WRITE Event\_Algorithm\_Inhibit = FALSE

[ In BTL Test Plan, modify the Test Conditionality in tests 7.3.1.X6.1, 7.3.1.X7.1 and 7.3.1.X7.2 ]

**5.2 Alarm and Event Management - Notification - Internal - B****5.2.1 Base Requirements**

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

<b>BTL - 7.3.1.10.1 - Event Enable Tests for TO_OFFNORMAL and TO_NORMAL</b>		
	<b>Test Conditionality</b>	If the IUT cannot be configured to meet the configuration requirements then this test shall be skipped.
	<b>Test Directives</b>	If Event Enrollment objects are supported, ensure this functionality is tested on Event Enrollment objects.
	<b>Testing Hints</b>	The BTL will apply this to a single object. The pretester should apply it to all objects that support alarm generation.
<b>135.1-2013 - 7.3.1.12 - Notify Type Test</b>		
	<b>Test Conditionality</b>	If the IUT cannot be configured to meet the 135.1-2013 configuration requirements then this test shall be skipped.

	<b>Test Directives</b>	If Event Enrollment objects are supported, ensure this functionality is also tested on Event Enrollment objects.
	<b>Testing Hints</b>	
<b>135.1-2013 - 8.4 - ConfirmedEventNotification Service Initiation Tests</b>		
	<b>Test Conditionality</b>	Must be executed unless IUT only supports 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.
	<b>Test Directives</b>	
	<b>Testing Hints</b>	
<b>135.1-2013 - 8.5 - UnconfirmedEventNotification Service Initiation Tests</b>		
	<b>Test Conditionality</b>	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.
	<b>Test Directives</b>	
	<b>Testing Hints</b>	
<b>BTL - 7.3.1.X9.1 [7.3.1.22.1] - Event Detection Enable Inhibits Event Generation</b>		
	<b>Test Conditionality</b>	If Protocol Revision < 13, then this test shall be skipped.
	<b>Test Directives</b>	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.
	<b>Testing Hints</b>	
<b>BTL - 7.3.1.X9.2 [7.3.1.22.2] - Event Detection Enable Inhibits FAULT</b>		
	<b>Test Conditionality</b>	If Protocol Revision < 13, then this test shall be skipped.
	<b>Test Directives</b>	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.
	<b>Testing Hints</b>	
<b>BTL - 7.3.1.X6.1 [7.3.1.19.1] - Event Algorithm Inhibit Test</b>		
	<b>Test Conditionality</b>	If the IUT has no object in which Apply this test when the Event_Algorithm_Inhibit property is present in an object which and does not support the Event_Algorithm_Inhibit_Ref property. If the IUT cannot be configured to contain such an object, then this test shall be skipped., 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.
	<b>Test Directives</b>	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.
	<b>Testing Hints</b>	
<b>BTL - 7.3.1.X7.1 [7.3.1.20.1] - Event Algorithm Inhibit Ref Test</b>		
	<b>Test Conditionality</b>	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.
	<b>Test Directives</b>	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.
	<b>Testing Hints</b>	

<b>BTL - 7.3.1.X7.2 [7.3.1.20.2] - Event Algorithm Inhibit Writable Test</b>		
	<b>Test Conditionality</b>	If the IUT has no Apply this test when the Event_Algorithm_Inhibit property is present in an object in which the Event_Algorithm_Inhibit_Ref property is absent or can be made uninitialized. If the IUT cannot be configured to contain such an object, then this test shall be skipped. or has no object in which Event_Detection_Enable can be made TRUE, this test shall be skipped.
	<b>Test Directives</b>	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.
	<b>Testing Hints</b>	
<b>BTL - 8.5.X9.10 - After FAULT-to-NORMAL, Re-Notification of OFFNORMAL</b>		
	<b>Test Conditionality</b>	If the IUT has no object in which CHANGE_OF_RELIABILITY is implemented in an object that can be configured into an offnormal state, this test shall be skipped.
	<b>Test Directives</b>	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 shall be selected.
	<b>Testing Hints</b>	

[ In BTL Checklist, remove the 27 instances in specific objects devoted to testing the Reliability\_Evaluation\_Inhibit property, and instead in BTL Test Plan, add a new section in Alarm and Event Management - Notification - Internal - B devoted to testing the Reliability\_Evaluation\_Inhibit property. ]

### 3 Objects

Support	Listing	Option
<Any object that contains this entry>		
	O	Contains an object with Reliability_Evaluation_Inhibit Property

### 5 Alarm and Event Management BIBBs

. . .

<b>Alarm and Event Management - Notification - Internal - B</b>		
	O	Implements the Reliability_Evaluation_Inhibit property
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.		

[ In BTL Test Plan, add a new section in Alarm and Event Management - Notification - Internal - B devoted to testing the Reliability\_Evaluation\_Inhibit property with tests 7.3.1.X8.1 and 7.3.1.X8.2 ]

## 5.2 Alarm and Event Management - Notification - Internal - B

### 5.2.X42 Implements the Reliability\_Evaluation\_Inhibit Property

The IUT supports the Reliability\_Evaluation\_Inhibit property to control the reliability evaluation in objects.

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. 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.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. 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.

[ In BTL Test Plan, remove the 27 existing sections within clause 3 which are testing the Reliability\_Evaluation\_Inhibit property in specific objects with the same tests 7.3.1.X8.1 and 7.3.1.X8.2 ]

### 3.## - 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 - Event Detection Enable Inhibits Event Generation		
	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	



## **BTL-16.1fix1-3: Clarify How to Read Large Properties - BTLWG-601 [CR-0413]**

### **Overview:**

CR-0413 response specified that in test 7.1.X3 to add a *Notes to Tester* mentioning that element-by-element reading of the properties may be necessary if the returned result for the whole property value would exceed the Maximum Transmissible APDU.

### **Changes:**

[ In BTL Specified Tests, in test 7.1.X3, add to the existing Notes to Tester. ]

### **7.1.X3 Verifying Property\_List against the EPICS**

Reason for Change: Addendum 135-2010ao-5. Additionally, CR-0413 response directed to add a Note to Tester mentioning that element-by-element reading of the properties may be necessary.

Purpose: To verify the correct content of the Property\_List using the properties in each object as claimed in the EPICS.

Test Concept: Match the properties in each object as claimed in the EPICS, against the content of each object's Property\_List.

Test Conditionality: If Protocol\_Revision is not present, or Protocol\_Revision < 14, then this test shall be skipped.

### **Test Steps:**

1. READ OL = Object\_List
2. REPEAT (O1, each object in the content of OL)
3. READ PL = Property\_List, in the selected object instance O1
4. CHECK (that the property identifiers in the EPICS for O1 and those in the Property\_List property match, except as specified in Notes to Tester)

Notes to Tester: Object\_Name (77), Object\_Type (79), Object\_Identifier (75), and Property\_List (371) will appear in the EPICS, but shall not appear in the Property\_List value. Any proprietary properties that are supported for the object-type shall be in the Property\_List. (see BACnet 15.7.3.1.2). The order in which property identifiers appear in the EPICS, is not required to match the order that they appear in the Property\_List value. If the whole BACnetARRAY cannot be read because it exceeds the Maximum Transmissible APDU, then the tester shall read it element-by-element in order to obtain the complete value.

## BTL-16.1fix1-4: Fix Out\_Of\_Service Accumulator Test - BTLWG-610 [CR-0463]

### Overview:

CR-0463 noted that Pulse\_Rate is optional and that 7.3.2.X37.1.6 Out\_Of\_Service Accumulator Test does not make an allowance for it.

### Changes:

[In BTL Specified Tests, modify clause 7.3.2.X37.1.6]

#### 7.3.2.X37.1.6 Out\_Of\_Service Accumulator Test

Reason for Change: New test for Accumulator object.

Purpose: This test case verifies that Present\_Value, Pulse\_Rate, and the Reliability property are writable when Out\_Of\_Service is TRUE.

Test Concept: Select one instance of each appropriate object type and test it as described. Verify the interrelationship between the Out\_Of\_Service, Status\_Flags, and Reliability 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. If the Reliability property is not supported then step 5 shall be omitted.

Test Steps:

1. IF (Out\_Of\_Service is writable) THEN  
    WRITE Out\_Of\_Service = TRUE  
ELSE  
    MAKE (Out\_Of\_Service TRUE)
2. VERIFY Out\_Of\_Service = TRUE
3. VERIFY Status\_Flags = (?, ?, ?, TRUE)
4. REPEAT X = (all values meeting the functional range requirements of 7.2.1) DO {  
    WRITE Present\_Value = X  
    VERIFY Present\_Value = X  
}
5. IF (Reliability is present and writable) THEN  
    REPEAT X = (all values of the Reliability enumeration appropriate to the object type except  
        NO\_FAULT\_DETECTED) DO {  
        WRITE Reliability = X  
        VERIFY Reliability = X  
        VERIFY Status\_Flags = (TRUE?, TRUE, ?, TRUE)  
        WRITE Reliability = NO\_FAULT\_DETECTED  
        VERIFY Reliability = NO\_FAULT\_DETECTED  
        VERIFY Status\_Flags = (?, FALSE, ?, TRUE)  
    }  
}
6. IF (the object has a Pulse\_Rate property) THEN {  
    REPEAT X = (all values meeting the functional range requirements of 7.2.1) DO {  
        WRITE Pulse\_Rate = X  
        VERIFY Pulse\_Rate = X  
    }  
}
7. IF (Out\_Of\_Service is writable) THEN  
    WRITE Out\_Of\_Service = FALSE  
ELSE  
    MAKE (Out\_Of\_Service FALSE)
8. VERIFY Out\_Of\_Service = FALSE
9. VERIFY Status\_Flags = (?, ?, ?, FALSE)

## BTL-16.1fix1-5: Fix Out\_Of\_Service Usage in Value Objects - BTLWG-737 [CR-0456]

### Overview:

Changes to 7.3.1.1 to deal with the CR-0456.

### Changes:

[In BTL Specified Tests, Add section 7.3.1.1]

#### 7.3.1.1 Out\_Of\_Service, Status\_Flags, and Reliability Tests

[In BTL Specified Tests, add a new test to section 7.3.1.1]

##### 7.3.1.1.XI Out\_Of\_Service, Status\_Flags, and Reliability TestsTest

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

BACnet Reference Clauses: 12.1.7, 12.1.9, 12.1.10, 12.2.7, 12.2.9, 12.2.10, 12.3.7, 12.3.9, 12.3.10, 12.4.6, 12.4.8, 12.4.9, 12.6.7, 12.6.9, 12.6.10, 12.7.7, 12.7.9, 12.7.10, 12.8.6, 12.8.8, 12.8.9, 12.15.8, 12.15.10, 12.15.11, 12.16.8, 12.16.10, 12.16.11, 12.17.6, 12.17.8, 12.17.9, 12.18.7, 12.18.9, 12.18.10, 12.19.7, 12.19.9, 12.19.10, 12.20.6, 12.20.8, 12.20.9, 12.23.7, 12.23.9, and 12.23.10.

Purpose: ~~This test case verifies that~~ *To verify that* Present\_Value is writable when Out\_Of\_Service is TRUE and ~~It also that the interrelationship between the Out\_Of\_Service, Status\_Flags, and Reliability properties. If the PICS indicates that the Out\_Of\_Service property of the object under test is not writable, and if the value of the property cannot be changed by other means, then this test shall be omitted. This test applies to Accumulator, Analog Input, Analog Output, Analog Value, Binary Input, Binary Output, Binary Value, Life Safety Point, Life Safety Zone, Multi state Input, Multi state Output, Multi state Value, Loop and Pulse Converter objects.~~

Test Concept: ~~The IUT will select one instance of each appropriate object type and test it as described. If the Reliability property is not supported then step 4 shall be omitted. The value of the Out\_Of\_Service property is set to TRUE and the Present\_Value property is tested to be writable. The value of the Status\_Flags property is validated and, if present, the value of the Reliability property is also validated. The value of the Status\_Flags property, SF1, and, if present, the Reliability property, R1, are checked to ensure they return to their initial values when the value of the Out\_Of\_Service property is set to FALSE.~~

Configuration Requirements: *If the selected object is commandable, the values of the entries in the Priority\_Array above the selected priority, PTY1, shall be NULL.*

### Test Steps:

1. SF1 = Status\_Flags
2. R1 = Reliability
34. IF (Out\_Of\_Service is writable) THEN
  - WRITE Out\_Of\_Service = TRUE
  - ELSE
    - MAKE (Out\_Of\_Service TRUE)
  - 42. VERIFY Out\_Of\_Service = TRUE
  - 53. VERIFY Status\_Flags = (?, FALSE?, ?, TRUE)
  - 64. REPEAT X = (all values meeting the functional range requirements of 7.2.1) DO {
    - WRITE Present\_Value, PTY1 = X
    - VERIFY Present\_Value = X
75. IF (Reliability is present and writable) THEN
  - REPEAT X = (all values of the Reliability enumeration appropriate to the object type except NO\_FAULT\_DETECTED) DO {
    - WRITE Reliability = X
    - VERIFY Reliability = X
    - VERIFY Status\_Flags = (?, TRUE, ?, TRUE)
    - WRITE Reliability = NO\_FAULT\_DETECTED

```

        VERIFY Reliability = NO_FAULT_DETECTED
        VERIFY Status_Flags = (?, FALSE, ?, TRUE)
    }
86. IF (Out_Of_Service is writable) THEN
    WRITE Out_Of_Service = FALSE
ELSE
    MAKE (Out_Of_Service FALSE)
97. VERIFY Out_Of_Service = FALSE
108. VERIFY Status_Flags = (?, ?, ?, FALSE) SFI
11. VERIFY Reliability = RI

```

**Notes to Tester:** If the object being tested is commandable and there is an internal process writing to the Present\_Value property, then each WriteProperty request shall contain a priority sufficient to override the internal process. After step 4 the priority array slot shall be relinquished.

[In BTL Specified Tests, add a new test to section 7.3.1.1]

#### 7.3.1.1.X2 Out\_Of\_Service for Commandable Value Objects Test

Purpose: To verify that Present\_Value is no longer updated by software local to the IUT when Out\_Of\_Service is TRUE.

Test Concept: Select an object who's Present\_Value is being modified by software local to the IUT at Priority PTY1. The value of the Out\_Of\_Service property is set to TRUE, the Present\_Value property is written at PTY1 and the Present\_value is checked to ensure the Present-Value is no longer being modified by software local to the IUT.

Configuration Requirements: The values of the entries in the Priority\_Array above PTY1 shall be NULL.

Test Steps:

1. MAKE (Present\_Value = PV1, any valid value, using software local to the IUT)
2. IF (Out\_Of\_Service is writable) THEN
  - WRITE Out\_Of\_Service = TRUE
  - ELSE
    - MAKE (Out\_Of\_Service TRUE)
3. VERIFY Present\_Value = PV1
4. WRITE Present\_Value, PTY1 = PV2, any valid value other than PV1
5. MAKE (Present\_Value = PV3, any valid value other than PV2, using software local to the IUT)
6. VERIFY Present\_Value = PV2

[In BTL Test Plan, change 3.1.2]

[In BTL Test Plan, change 3.2.3]

[In BTL Test Plan, change 3.5.2]

[In BTL Test Plan, change 3.6.3]

[In BTL Test Plan, change 3.13.2]

[In BTL Test Plan, change 3.14.2]

[In BTL Test Plan, change 3.15.3]

[In BTL Test Plan, change 3.39.2]

[In BTL Test Plan, change 3.40.2]

[In BTL Test Plan, change 3.42.3]

[In BTL Test Plan, change 3.44.2]

[In BTL Test Plan, change 3.45.2]

[In BTL Test Plan, change 3.49.2]

[In BTL Test Plan, change 3.53.2]

[In BTL Test Plan, change 3.54.4]

[In BTL Test Plan, change 3.55.3]

The Out\_Of\_Service property in this object type is writable.

<b>BTL - 7.3.1.1.X1 - Out_Of_Service, Status_Flags, and Reliability Test</b>
--

	<b>Test Conditionality</b>	Must be executed.
	<b>Test Directives</b>	
	<b>Testing Hints</b>	

[In BTL Test Plan, change 3.3.2]

[In BTL Test Plan, change 3.7.2]

[In BTL Test Plan, change 3.16.2]

[In BTL Test Plan, change 3.24.2]

[In BTL Test Plan, change 3.25.2]

[In BTL Test Plan, change 3.26.2]

[In BTL Test Plan, change 3.27.2]

[In BTL Test Plan, change 3.28.2]

[In BTL Test Plan, change 3.29.2]

[In BTL Test Plan, change 3.30.2]

[In BTL Test Plan, change 3.31.2]

[In BTL Test Plan, change 3.32.2]

[In BTL Test Plan, change 3.33.2]

[In BTL Test Plan, change 3.34.2]

[In BTL Test Plan, change 3.35.2]

The Out\_Of\_Service property in this object type is writable.

<b>BTL - 7.3.1.1.X1 - Out_Of_Service, Status_Flags, and Reliability Test</b>		
	<b>Test Conditionality</b>	Must be executed.
	<b>Test Directives</b>	
	<b>Testing Hints</b>	
<b>BTL - 7.3.1.1.X2 - Out_Of_Service for Commandable Value Objects Test</b>		
	<b>Test Conditionality</b>	If the object is commandable, this test must be executed.
	<b>Test Directives</b>	
	<b>Testing Hints</b>	

**BTL-16.1fix1-6: Fix Reliability\_Evaluation\_Inhibit Tests - BTLWG-736 & BTLWG-741 [CR-0455]****Overview:**

This document contains the changes required by CR-0455 which noted that testing of Reliability\_Evaluation\_Inhibit incorrectly relied on the IUT supporting event reporting.

**Changes:**

[ Modify all BTL Test Plan entries for 7.3.1.X8.1 and 7.3.1.X8.2 ]

<b>BTL - 7.3.1.X8.1 - Reliability_Evaluation_Inhibit Test</b>		
	<b>Test Conditionality</b>	If no object exists in the IUT for which fault conditions can be generated detected and which supports a modifiable Reliability_Evaluation_Inhibit property, then this test shall be skipped.
	<b>Test Directives</b>	
	<b>Testing Hints</b>	
<b>BTL - 7.3.1.X8.2 - Reliability_Evaluation_Inhibit Summarization Test</b>		
	<b>Test Conditionality</b>	If no object exists in the IUT for which fault conditions can be generated detected, which supports a modifiable Reliability_Evaluation_Inhibit property and which supports intrinsic reporting then this test shall be skipped.
	<b>Test Directives</b>	
	<b>Testing Hints</b>	

[ Modify 7.3.1.X8.1 in BTL Specified Tests ]

**7.3.1.X8.1 Reliability\_Evaluation\_Inhibit Test**

Reason for Change: New functionality added with Addendum 135-2010af. This test does not exist in 135.1-2013.

Purpose: To verify that Reliability\_Evaluation\_Inhibit controls whether or not fault conditions are detected.

Test Concept: Select an event generating object, O1, which supports the Reliability\_Evaluation\_Inhibit property. With Reliability\_Evaluation\_Inhibit FALSE, make a fault condition exist. Verify that Reliability changes and, if event reporting is supported, that a notification is generated. Set Reliability\_Evaluation\_Inhibit to TRUE. Verify that the Reliability changes to NO\_FAULT\_DETECTED and, if event reporting is supported, that a TO\_NORMAL notification is generated. Remove the fault condition and ensure that no notification is generated. Make a fault condition exist and verify that Reliability remains NO\_FAULT\_DETECTED, and that no notification is generated.

Test Configuration: O1 is configured to detect and, if event reporting is supported, report unconfirmed events, is in the NORMAL state, and Reliability\_Evaluation\_Inhibit equals FALSE, so that reliability evaluation for that object is configured to detect fault conditions. If no object exists in the IUT for which fault conditions can be generated then this test shall be skipped.

**Test Steps:**

1. VERIFY Event\_State = NORMAL
2. VERIFY Reliability = NO\_FAULT\_DETECTED
3. MAKE(a fault condition exist for O1)
4. IF the IUT supports event reporting THEN
  - BEFORE Notification Fail Time
  - RECEIVE UnconfirmedEventNotification-Request
    - 'Process Identifier' = (the value configured for the transition),
    - 'Initiating Device Identifier' = IUT,
    - 'Event Object Identifier' = O1,
    - 'Time Stamp' = (any valid timestamp),
    - 'Priority' = (any valid priority),
    - 'Event Type' = CHANGE\_OF\_RELIABILITY,
    - 'Notify Type' = ALARM | EVENT,

'Message Text' =	(any valid message text),
'AckRequired' =	TRUE   FALSE,
'From State' =	NORMAL,
'To State' =	FAULT,
'Event Values' =	(any values appropriate to CHANGE_OF_RELIABILITY)

5. VERIFY Reliability <> NO\_FAULT\_DETECTED
6. IF Reliability\_Evaluation\_Inhibit is writable THEN  
     WRITE Reliability\_Evaluation\_Inhibit = TRUE  
   ELSE  
     MAKE(Reliability\_Evaluation\_Inhibit TRUE)
7. IF the IUT supports event reporting THEN  
     BEFORE **Internal Processing Fail Time + Notification Fail Time**  
       RECEIVE UnconfirmedEventNotification-Request  
         'Process Identifier' = (the value configured for the transition),  
         'Initiating Device Identifier' = IUT,  
         'Event Object Identifier' = O1,  
         'Time Stamp' = (any valid timestamp),  
         'Priority' = (any valid priority),  
         'Event Type' = CHANGE\_OF\_RELIABILITY,  
         'Notify Type' = ALARM | EVENT,  
         'Message Text' = (any valid message text),  
         'AckRequired' = TRUE | FALSE,  
         'From State' = FAULT,  
         'To State' = NORMAL,  
         'Event Values' = (any values appropriate to CHANGE\_OF\_RELIABILITY)
8. VERIFY Reliability = NO\_FAULT\_DETECTED
9. VERIFY Event\_State = NORMAL
10. MAKE(remove the fault condition)
11. WAIT **Notification Fail Time**
12. CHECK (that the IUT did not send any event notifications for O1)
13. VERIFY Reliability = NO\_FAULT\_DETECTED
14. MAKE(a fault condition exist for O1)
15. WAIT **Notification Fail Time**
16. VERIFY Reliability = NO\_FAULT\_DETECTED
17. VERIFY Event\_State = NORMAL
18. CHECK (that the IUT did not send any event notifications for O1)

Notes to Tester: This behavior can alternately be tested using the ConfirmedEventNotification service, but it is not necessary to test both.

## **BTL-16.1fix1-7: Fix Adjust\_Value Write Test - BTLWG-751 [CR-0459]**

### **Overview:**

Clarification Request BTL-CR-0459 identified a problem in the Adjust\_Value Write Test. This proposal addresses that and other concerns with the test.

### **Changes:**

[Modify test 7.3.2.X38.1.1 in BTL Interim Tests]

#### **7.3.2.X38.1.1 Adjust\_Value Write Test**

**Reason for Change:** No test exists for this functional requirement. There is no SSPC proposal for this change.

**Purpose:** To verify the correct write operation of a Pulse Converter's several properties, when writing the Adjust\_Value. Count\_Before\_Change reflects the prior Count before a write to the Adjust\_Value property.

**Test Configuration:** Select a Pulse Converter object for which the pulse can be stopped so that Count remains unchanged during the test.

#### **Test Steps:**

[Note that steps 1,3-7 were deleted]

1. READ OldC = Count
2. WRITE Adjust\_Value = (NewA, any valid value)
3. VERIFY Count = OldC - (NewA / Scale\_Factor)
4. VERIFY Present\_Value = Count \* Scale\_Factor
5. VERIFY Count\_Change\_Time  $\approx$  (the current local time)
6. VERIFY Count\_Before\_Change = OldC



**BTL-16.1fix1-8: Correct Test Plan Entry for AE-N-E-B - BTLWG-880****Overview:**

The test plan entry for 8.5.X9.10 in AE-N-E-B is incorrect. When executed for AE-N-E-B, test 8.5.X9.10 is expected to be run using an Event\_Enrollment object monitoring an object in another device. Test coverage for local objects is provided in AE-N-I-B.

**Changes:****5.3 Alarm and Event Management - Notification - External - B****5.3.1 Base Requirements**

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

<b>BTL - 8.5.X9.10 - After FAULT-to-NORMAL, Re-Notification of OFFNORMAL</b>		
	<b>Test Conditionality</b>	If the IUT has no object in which CHANGE_OF_RELIABILITY is implemented and which can be configured into an offnormal state, this test shall be skipped. If the IUT does not support an Event Enrollment object which is capable of generating OFFNORMAL transitions, this test shall be skipped.
	<b>Test Directives</b>	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 shall be selected. Execute test using an Event Enrollment object monitoring an object (O1) in a device other than the IUT.
	<b>Testing Hints</b>	

## **BTL-16.1fix1-9: COV Subscription Test Changes - BTLWG-841 [CR-0458]**

### **Overview:**

In response to CR-0458, the test should be modified by adding a step after step 4:

5. MAKE (the IUT stop resubscribing, if it resubscribes automatically)

### **Changes:**

[In the Interim Tests document, add the following section into each TP section covering commandability in objects]

### **8.11.X1.3 Change of Value Notification Arrives after Subscription has Expired**

Reason for Change: New test to support DS-COVP-A testing

Purpose: To verify that an appropriate error is returned if a COV notification arrives after the subscription time period has expired.

Test Concept: A subscription for COV notifications is established, using a Lifetime of L. L shall be set to a value less than 24 hours and large enough to complete the test.

Test Steps:

1. MAKE (the IUT send a SubscribeCOVProperty-Request),
2. RECEIVE SubscribeCOVProperty-Request  
'Subscriber Process Identifier' = (any valid process identifier),  
'Monitored Object Identifier' = X  
'Issue Confirmed Notifications' = TRUE,  
'Lifetime' = L,  
'Monitored Property Identifier' = (the property Y to be monitored), BACnet Testing Laboratories - Specified Tests  
'COV Increment' = (Any REAL value -- optional)
3. TRANSMIT BACnet-SimpleACK-PDU
4. BEFORE Notification Fail Time  
TRANSMIT ConfirmedCOVNotification-Request,  
'Subscriber Process Identifier' = (the process identifier used in step 1),  
'Initiating Device Identifier' = TD,  
'Monitored Object Identifier' = X  
'Time Remaining' = (any value appropriate for the Lifetime selected),  
'List of Values' = (values appropriate to the property Y subscribed to, and any other properties the IUT provides with it, such as Status\_Flags)  
RECEIVE BACnet-SimpleACK-PDU
5. MAKE (the IUT stop resubscribing, if it resubscribes automatically)
6. WAIT (a value two times Lifetime)
7. TRANSMIT ConfirmedCOVNotification-Request,  
'Subscriber Process Identifier' = (the process identifier used in step 1),  
'Initiating Device Identifier' = TD,  
'Monitored Object Identifier' = X  
'Time Remaining' = (any value appropriate for the Lifetime selected),  
'List of Values' = (values appropriate to the property Y subscribed to, and any other properties the IUT provides with it, such as Status\_Flags)
8. IF (Protocol\_Revision is present and Protocol\_Revision  $\geq$  10) THEN  
RECEIVE BACnet-Error-PDU,  
Error Class = SERVICES,  
Error Code = (UNKNOWN\_SUBSCRIPTION) |  
(BACnet-SimpleACK-PDU)  
ELSE  
RECEIVE BACnet-Error-PDU,  
Error Class = SERVICES,  
Error Code = (any valid error code for class SERVICES) |

(BACnet-SimpleACK-PDU)

## BTL-16.1fix1-10: Update Conditionality for Event\_Detection\_Enable Test - BTLWG-653 [CR-0448]

### Overview:

Clarification Request 0448, pointed out that the Event\_Detection\_Enable Inhibits FAULT test cannot be run if the IUT does not support fault detection.

### Changes:

[In BTL Test Plan, modify AE-N-I-B entry for 7.3.1.X9 in the Base Requirements section]

BTL - 7.3.1.X9.2 - Event_Detection_Enable Inhibits FAULT		
	<b>Test Conditionality</b>	If Protocol_Revision < 13 <i>or if the IUT doesn't contain any event generating objects which support fault detection</i> , then this test shall be skipped.
	<b>Test Directives</b>	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.
	<b>Testing Hints</b>	

## BTL-16.1fix1-11: Improve Change of Value Notification Test Names - BTLWG-732

### Overview:

Rename tests to remove object types from title and within test function.

### Changes:

#### New Test Names:

**8.2.1 Change of Value Notification ~~from an Analog Input, Analog Output, and Analog Value Object's~~ *for Changes to Present\_Value Property in Objects with a COV\_Increment***

**8.2.3 Change of Value Notification ~~from a Binary Input, Binary Output, and Binary Value Object's~~ *for Changes to Present\_Value Property in Objects without a COV\_Increment***

**8.2.5 DELETE TEST** duplicate of 8.2.3.

**8.2.2 Change of Value Notification ~~from an Analog Input, Analog Output, and Analog Value Object's~~ *for Changes to Status\_Flags Property***

**8.2.4 DELETE Test** duplicate of 8.2.2

**8.2.6 DELETE Test** duplicate of 8.2.2

**8.3.1 Change of Value Notification ~~from an Analog Input, Analog Output, and Analog Value~~ *for Changes to Object's Present\_Value Property in Objects with a COV\_Increment***

**8.3.2 Change of Value Notification ~~from an Analog Input, Analog Output, and Analog Value Object's~~ *for Changes to Status\_Flags Property***

**8.3.3 Change of Value Notification ~~from a Binary Input, Binary Output, and Binary Value Object's~~ *for Changes to Present\_Value in Objects without a COV\_Increment* ~~Property~~**

**8.3.4 DELETE**

**8.3.5 DELETE**

**8.3.6 DELETE**

[In BTL Test Plan, modify all references to tests 8.2.1, 8.2.2, 8.3.1, and 8.3.2 in the following sections to match new test names shown above]

4.10.4 Supports COV for Analog Input Objects

4.10.5 Supports COV for Analog Output Objects

4.10.6 Supports COV for Analog Value Objects

4.10.21 Supports COV for Integer Value Objects

4.10.22 Supports COV for Large Analog Value Objects

4.10.23 Supports COV for Positive Integer Value Objects

4.10.32 Supports COV for Lighting Output Objects

[In BTL Test Plan, change test title for 8.2.3, change test reference for 8.2.4 to 8.2.2 and test title, change test title for 8.3.3, and change test reference of 8.3.4 to 8.3.2 in the following sections]

4.10.7 Supports COV for Binary Input Objects

4.10.8 Supports COV for Binary Output Objects

4.10.9 Supports COV for Binary Value Objects

4.10.33 Supports COV for Binary Lighting Output Objects

[In BTL Test Plan, Change test references for 8.2.5 to 8.2.3 and change test title, Change test title for tests 8.2.6 and 8.3.5, Change test references for 8.3.6 to 8.3.2 and change test title]

4.10.10 Supports COV for Life Safety Point Objects

4.10.11 Supports COV for Life Safety Zone Objects

4.10.13 Supports COV for Multi-state Input Objects

4.10.14 Supports COV for Multi-state Output Objects

4.10.15 Supports COV for Multi-state Value Objects

- 4.10.16 Supports COV for CharacterString Value Objects
- 4.10.17 Supports COV for Date Value Objects
- 4.10.18 Supports COV for Date Pattern Value Objects
- 4.10.19 Supports COV for DateTime Value Objects
- 4.10.20 Supports COV for DateTime Pattern Value Objects
- 4.10.24 Supports COV for Time Value Objects
- 4.10.25 Supports COV for Time Pattern Value Objects
- 4.10.26 Supports COV for OctetString Value Objects

[In BTL Specified Tests, update the following tests with new names and content]

## 8.2 ConfirmedCOVNotification Service Initiation Tests

### 8.2.1 Change of Value Notification ~~from an Analog Input, Analog Output, and Analog Value Object's~~ *for Changes to Present\_Value Property in Objects with a COV\_Increment*

Reason for Change: ~~Add more primitive value objects.~~ Updated description of the 'List of Values' to improve readability. Updated 'Configuration Requirements'. Add clarification to test that the last COVNotification shall reflect the correct values. *Improved test name and wording to include generic object references.*

Purpose: To verify that the IUT can initiate ConfirmedCOVNotification service requests conveying a change of the Present\_Value property ~~of Analog Input, Analog Output, and Analog Value~~ in objects *that support COV\_Increment*.

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

Configuration Requirements: At the beginning of the test, the Out\_Of\_Service property shall have a value of FALSE. *Select an object where Present\_Value is not expected to change outside the tester's control by more than COV\_Increment or which has a writable Out\_Of\_Service.*

*Notes to Tester: The IUT may initiate additional COVNotifications. The final COVNotification shall accurately reflect Present\_Value and Status\_Flags.*

Test Steps:

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

1. TRANSMIT SubscribeCOV-Request,
 

'Subscriber Process Identifier' =	(any value > 0 chosen by the TD),
'Monitored Object Identifier' =	X,
'Issue Confirmed Notifications' =	TRUE,
'Lifetime' =	L
2. RECEIVE BACnet-SimpleACK-PDU
3. BEFORE **Notification Fail Time**

RECEIVE ConfirmedCOVNotification-Request,

'Subscriber Process Identifier' =	(the same value used in step 1),
'Initiating Device Identifier' =	IUT,
'Monitored Object Identifier' =	X,
'Time Remaining' =	(any value appropriate for the Lifetime selected),
'List of Values' =	(the initial Present_Value and initial Status_Flags)
4. TRANSMIT BACnet-SimpleACK-PDU
5. TRANSMIT ReadProperty-Request,
 

'Object Identifier' =	X,
'Property Identifier' =	COV_Increment
6. RECEIVE BACnet-ComplexACK-PDU,
 

'Object Identifier' =	X,
-----------------------	----

```

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

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

```

### 8.2.2 Change of Value Notification ~~from an Analog Input, Analog Output, and Analog Value Object's~~ **for Changes to Status\_Flags Property**

Reason for Change: ~~Add more primitive value objects.~~ Updated 'Configuration Requirements'. Removed extraneous SimpleACKs after WRITE statements. Updated descriptive text for 'List of Value' property. Modified to make generic to all object types.

Purpose: To verify that the IUT can initiate ConfirmedCOVNotification service requests conveying a change of the Status\_Flags property of ~~Analog Input, Analog Output and Analog Value~~ objects.

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

Configuration Requirements: At the beginning of the test, the Out\_Of\_Service property shall have a value of FALSE. *Select an object where Present Value is not expected to change outside the tester's control by more than COV\_Increment, if COV\_Increment is supported or which has a writable Out\_Of\_Service.*

Test Steps:

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

1. TRANSMIT SubscribeCOV-Request,  
     'Subscriber Process Identifier' = (any value > 0 chosen by the TD),  
     'Monitored Object Identifier' = X,  
     'Issue Confirmed Notifications' = TRUE,  
     'Lifetime' = L
2. RECEIVE BACnet-SimpleACK-PDU
3. BEFORE **Notification Fail Time**  
     RECEIVE ConfirmedCOVNotification-Request,  
     'Subscriber Process Identifier' = (the same value used in step 1),  
     'Initiating Device Identifier' = IUT,  
     'Monitored Object Identifier' = X,  
     'Time Remaining' = (any value appropriate for the Lifetime selected),  
     'List of Values' = (the initial Present\_Value and initial Status\_Flags)
4. TRANSMIT BACnet-SimpleACK-PDU
5. WRITE X, Out\_Of\_Service = TRUE | WRITE X, Status\_Flags = (a value that differs from initial Status\_Flags) |  
     MAKE (Status\_Flags = any value that differs from initial Status\_Flags)
- ~~6. IF (WriteProperty is used in step 5) THEN~~  
     ~~RECEIVE BACnet SimpleACK PDU~~
76. BEFORE **Notification Fail Time**  
     RECEIVE ConfirmedCOVNotification-Request,  
     'Subscriber Process Identifier' = (the same value used in step 1),  
     'Initiating Device Identifier' = IUT,  
     'Monitored Object Identifier' = X,  
     'Time Remaining' = (any value appropriate for the Lifetime selected),  
     'List of Values' = ~~(the initial~~the current Present\_Value and new Status\_Flags)
87. TRANSMIT BACnet-SimpleACK-PDU
98. TRANSMIT SubscribeCOV-Request,  
     'Subscriber Process Identifier' = (the same value used in step 1),  
     'Monitored Object Identifier' = X
- ~~109. RECEIVE BACnet-SimpleACK-PDU~~
- ~~110. IF (Out\_Of\_Service was changed in step 5) THEN~~  
     ~~WRITE X, Out\_Of\_Service = FALSE~~  
     ~~RECEIVE BACnet SimpleACK PDU~~

### 8.2.3 Change of Value Notification ~~from a Binary Input, Binary Output, and Binary Value Object's~~ *for Changes to Present\_Value Property in Objects without a COV\_Increment*

Reason for Change: Updated the 'Configuration Requirements'. Removed extraneous SimpleACKs that appear after WRITE statements. Modified descriptive text for 'List of Values' properties. Add clarification to test that the last COVNotification shall reflect the correct values. Modified test name and test to accept generic objects that do not support the COV\_Increment property.

Purpose: To verify that the IUT can initiate ConfirmedCOVNotification service requests conveying a change of the Present\_Value property of ~~Binary Input, Binary Output, and Binary Value~~ objects *that do not support COV\_Increment*.

Test Concept: A subscription for COV notifications is established, using a Lifetime of L. L shall be set to a value less than 24 hours and large enough to complete the test. The Present\_Value of the monitored object is changed and a notification shall be received. The Present\_Value may be changed using the WriteProperty service or by another means such as changing the



input signal represented by a Binary Input object. For some implementations it may be necessary to write to the Out\_Of\_Service property first to accomplish this task. For implementations where it is not possible to write to these properties at all the vendor shall provide an alternative trigger mechanism to accomplish this task. All of these methods are equally acceptable.

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

*Notes to Tester: The IUT may initiate additional COVNotifications. The final COVNotification shall accurately reflect Present\_Value and Status\_Flags.*

Test Steps:

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

1. TRANSMIT SubscribeCOV-Request,  
     'Subscriber Process Identifier' = (any value > 0 chosen by the TD),  
     'Monitored Object Identifier' = X,  
     'Issue Confirmed Notifications' = TRUE,  
     'Lifetime' = L
2. RECEIVE BACnet-SimpleACK-PDU
3. BEFORE **Notification Fail Time**  
     RECEIVE ConfirmedCOVNotification-Request,  
     'Subscriber Process Identifier' = (the same value used in step 1),  
     'Initiating Device Identifier' = IUT,  
     'Monitored Object Identifier' = X,  
     'Time Remaining' = (any value appropriate for the Lifetime selected),  
     'List of Values' = (the initial Present\_Value and initial Status\_Flags)
4. TRANSMIT BACnet-SimpleACK-PDU
5. IF (Out\_Of\_Service is writable) THEN  
     WRITE X, Out\_Of\_Service = TRUE  
     BEFORE **Notification Fail Time**  
     RECEIVE ConfirmedCOVNotification-Request,  
     'Subscriber Process Identifier' = (the same value used in step 1),  
     'Initiating Device Identifier' = IUT,  
     'Monitored Object Identifier' = X,  
     'Time Remaining' = (any value appropriate for the Lifetime selected),  
     'List of Values' = ~~(the initial~~ReportedPV = the current Present\_Value, and new Current  
     Status\_Flags)  
     TRANSMIT BACnet-SimpleACK-PDU
6. IF (Present\_Value is now writable) THEN  
     WRITE X, Present\_Value = (any value that differs from ~~"initial Present\_Value"~~ ReportedPV)  
     ELSE  
     MAKE (Present\_Value = any value that differs from ~~"initial Present\_Value"~~ ReportedPV)
7. BEFORE Notification Fail Time  
     RECEIVE ConfirmedCOVNotification-Request,  
     'Subscriber Process Identifier' = (the same value used in step 1),  
     'Initiating Device Identifier' = IUT,  
     'Monitored Object Identifier' = X,  
     'Time Remaining' = (any value appropriate for the Lifetime selected),  
     'List of Values' = (the new Present\_Value and ~~new-Current~~ Status\_Flags)
8. TRANSMIT BACnet-SimpleACK-PDU
9. TRANSMIT SubscribeCOV-Request,  
     'Subscriber Process Identifier' = (the same value used in step 1),  
     'Monitored Object Identifier' = X
10. RECEIVE BACnet-SimpleACK-PDU
11. IF (Out\_Of\_Service is writable) THEN  
     WRITE X, Out\_Of\_Service = FALSE  
     ~~RECEIVE BACnet-SimpleACK-PDU~~

### 8.3 UnconfirmedCOVNotification Service Initiation Tests

#### 8.3.1 Change of Value Notification ~~from an Analog Input, Analog Output, and Analog Value~~ for Changes to **Object's Present\_Value Property in Objects with a COV\_Increment**

Reason for Change: Addendum 135-2008w-1, and 135-2-1-i-1 Add more primitive value objects and the Lighting Output Object. Add clarification to test that the last COVNotification shall reflect the correct values.

Purpose: To verify that the IUT can initiate UnconfirmedCOVNotification service requests conveying a change of the Present\_Value property of **Analog Input, Analog Output and Analog Value** objects.

Test Steps: The steps for this test case are identical to the test steps in 8.2.1 except that the SubscribeCOV service request in step 1 shall have a value of FALSE for the 'Issue Confirmed Notifications' parameter, all of the ConfirmedCOVNotification requests shall be UnconfirmedCOVNotification requests, and there is no acknowledgment of the unconfirmed services. The MAC address used for the notification message shall be such that the TD is one of the recipients.

*Notes to Tester: The IUT may initiate additional COVNotifications. The final COVNotification shall accurately reflect Present\_Value and Status\_Flags.*

#### 8.3.2 Change of Value Notification ~~from an Analog Input, Analog Output, and Analog Value~~ **Object's** for Changes to **Status\_Flags Property**

Reason for Change: Addendum 135-2008w-1 Add more primitive value objects.

Purpose: To verify that the IUT can initiate UnconfirmedCOVNotification service requests conveying a change of the Status\_Flags property of **Analog Input, Analog Output and Analog Value** objects.

Test Steps: The steps for this test case are identical to the test steps in 8.2.2 except that the SubscribeCOV service request in step 1 shall have a value of FALSE for the 'Issue Confirmed Notifications' parameter, all of the ConfirmedCOVNotification requests shall be UnconfirmedCOVNotification requests, and there is no acknowledgment of the unconfirmed services. The MAC address used for the notification message shall be such that the TD is one of the recipients.

#### 8.3.3 Change of Value Notification ~~from a Binary Input, Binary Output, and Binary Value~~ **Object's** for Changes to **Present\_Value in Objects without a COV\_IncrementProperty**

Reason for Change: Renamed test. Add clarification to test that the last COVNotification shall reflect the correct values.

Purpose: To verify that the IUT can initiate UnconfirmedCOVNotification service requests conveying a change of the Present\_Value property of **Binary Input, Binary Output, and Binary Value** objects that do not support COV\_Increment.

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

*Notes to Tester: The IUT may initiate additional COVNotifications. The final COVNotification shall accurately reflect Present\_Value and Status\_Flags.*