BACnet® TESTING LABORATORIES ADDENDA

Addendum as to BTL Test Package 16.1

Revision 5
Revised September 30, 2020

Approved by the BTL Working Group on July 13, 2020.
Approved by the BTL Working Group Voting Members on September 30, 2020.
Published on October 1, 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.

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 ~~strikethrough~~. 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-16.1as-1: - Add Value Source Information Testing - BTLWG-593

Overview:

Currently, the BTL Test Plan does not provide a test for conveying and recording the source of a write or command. In PR17, Value source mechanism is added for all commandable objects. Added testing for Value source mechanism and COVP testcases

Testing for COVM of Value_Source properties is covered in addendum Add-16.1aq.

Changes:

[In BTL Checklist add into each of the following object types, a "Supports Value Source Mechanism" option]
[Analog Output - 3.2 Supports the value source mechanism]
[Analog Value - 3.3 Supports the value source mechanism]
[Binary Output - 3.6 Supports the value source mechanism]
[Binary Value - 3.7 Supports the value source mechanism]
[Command - 3.9 Supports the value source mechanism]
[Multi-state Output - 3.15 Supports the value source mechanism]
[Multi-state Value - 3.16 Supports the value source mechanism]
[BitString Value - 3.24 Supports the value source mechanism]
[CharacterString Value - 3.25 Supports the value source mechanism]
[Date Pattern Value - 3.26 Supports the value source mechanism]
[Date Value - 3.27 Supports the value source mechanism]
[DateTime Pattern Value - 3.28 Supports the value source mechanism]
[DateTime Value - 3.29 Supports the value source mechanism]
[Integer Value - 3.30 Supports the value source mechanism]
[Large Analog Value - 3.31 Supports the value source mechanism]
[OctetString Value - 3.32 Supports the value source mechanism]
[Positive Integer Value - 3.33 Supports the value source mechanism]
[Time Pattern Value - 3.34 Supports the value source mechanism]
[Time Value - 3.35 Supports the value source mechanism]
[Life Safety Point - 3.39 Supports the value source mechanism]
[Life Safety Zone - 3.40 Supports the value source mechanism]
[Access Door - 3.42 Supports the value source mechanism]
[Load Control - 3.43 Supports the value source mechanism]
[Channel - 3.53 Supports the value source mechanism]
[Lighting Output - 3.54 Supports the value source mechanism]
[Binary Lighting Output - 3.55 Supports the value source mechanism]

	O	Supports the value source mechanism.

[Add option into DS-COVP-A]

Data Sharing - Change Of Value Property - A		
		…..
	O[3]	Can subscribe to Value_Source properties
[1] At least one of these options is required in order to claim conformance to this BIBB. [2] At least one of these options is required in order to claim conformance to this BIBB. [3] At least one of these options is required in order to claim conformance to this BIBB.		

[Add option into DS-COVP-B]

Data Sharing - Change Of Value Property - B		
		……..
	C[2]	Supports COVP to Value_Source properties
[1] At least one of these options is required in order to claim conformance to this BIBB. [2] At least one of these options is required in order to claim conformance to this BIBB.		

[In BTL Test Plan, add Value source mechanism tests into objects with required commandability]
[Analog Output, Binary Output, Multistate Output, Access Door, Lighting Output Object, Binary Lighting Output]

3.X.Y Supports the Value Source Mechanism

The IUT supports the Value Source Mechanism in <object>

BTL - 7.3.1.X42.Y3 - Value_Source Property None Test		
	Test Conditionality	Must be executed
	Test Directives	
	Testing Hints	
BTL - 7.3.1.X42.Y4 - Commandable Value Source Test		
	Test Conditionality	Must be executed
	Test Directives	
	Testing Hints	
BTL - 7.3.1.X42.Y1 - Writing to the Value_Source Property by a Device Other than the Device that Commanded the Property.		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

[In BTL Test Plan, add Value source mechanism tests into objects with optional commandability]

3.X.Y Supports the Value Source Mechanism

The IUT supports the Value Source Mechanism in <object>

BTL - 7.3.1.X42.Y2 - Non-commandable Value_Source Property Test		
	Test Conditionality	Must be executed if the Value Source Mechanism is supported in a instance where Present_Value is not commandable
	Test Directives	
	Testing Hints	
BTL - 7.3.1.X42.Y3 - Value_Source Property None Test		
	Test Conditionality	Must be executed if the Value Source Mechanism is supported in an instance where Present_Value is commandable
	Test Directives	
	Testing Hints	
BTL - 7.3.1.X42.Y4 - Commandable Value Source Test		
	Test Conditionality	Must be executed if the Value Source Mechanism is supported in an instance where Present_Value is commandable
	Test Directives	
	Testing Hints	
BTL - 7.3.1.X42.Y1 - Writing to the Value_Source Property by a Device Other than the Device that Commanded the Property.		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

[In BTL Test Plan, add Value source mechanism tests into non-commandable objects, excluding life safety objects]

3.X.Y Supports the Value Source Mechanism

The IUT supports the Value Source Mechanism in <object>

BTL - 7.3.1.X42.Y2 - Non-commandable Value_Source Property Test		
	Test Conditionality	Must be executed.

	Test Directives	
	Testing Hints	
BTL - 7.3.1.X42.Y1 - Writing to the Value_Source Property by a Device Other than the Device that Commanded the Property.		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

[In BTL Test Plan, add Value source mechanism tests into life safety objects]

3.X.Y Supports the Value Source Mechanism

The IUT supports the Value Source Mechanism in <object>

BTL - 7.3.1.X42.Y5 - Life Safety Value_Source Property Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 7.3.1.X42.Y1 - Writing to the Value_Source Property by a Device Other than the Device that Commanded the Property.		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

[In BTL Test Plan, add into "Supports Minimum Off_Time" sections: 3.6.7 Binary Output, 3.7.7 Binary Value,

BTL - 7.3.1.6.11 - Minimum_Off_Time - Value Source Mechanism		
	Test Conditionality	Must be executed if the IUT claims Value source mechanism.
	Test Directives	
	Testing Hints	

[In BTL Test Plan, add into " Supports Minimum_On_Time" sections: 3.6.8 Binary Output, 3.7.8 Binary Value]

BTL - 7.3.1.6.12 - Minimum_On_Time - Value Source Mechanism		
	Test Conditionality	Must be executed if the IUT claims Value source mechanism.
	Test Directives	
	Testing Hints	

[In BTL Test Plan, add into section 4.19 DS-COVP-A]

4.19.X Can Subscribe to Value_Source Properties

The IUT supports change of value notifications on Value_Source properties.

BTL - 8.11.X1.1 - Change of Value Notifications		
	Test Conditionality	Must be executed.
	Test Directives	Execute test using 'Monitored Property Identifier' = Value_Source
	Testing Hints	

[In BTL Test Plan, add into section 4.20 DS-COVP-B]

4.20.X Supports COVP to Value_Source Properties

The IUT supports change of value notifications on Value_Source properties.

BTL - 9.11.1.X11 - Confirmed Change of Value Notification from Property Value		
	Test Conditionality	Must be executed.

	Test Directives	Apply to at least 1 Value_Source property. Ensure that after all applications of this test (regardless of the property or datatype it is applied to), that the test has been applied at least once to each object type which supports COVP on one or more of its properties.
	Testing Hints	
BTL - 9.11.1.X12 - Unconfirmed Change of Value Notification from Property Value		
	Test Conditionality	Must be executed.
	Test Directives	Apply to at least 1 Value_Source property. Ensure that after all applications of this test (regardless of the property or datatype it is applied to), that the test has been applied at least once to each object type which supports COVP on one or more of its properties.
	Testing Hints	

[In BTL Specified Tests add new tests as shown.]

7.3.1.X42.Y1 Writing to the Value_Source Property by a Device Other than the Device that Commanded the Object.

Purpose: To verify the IUT correctly refuses an attempt to write a Value_Source property by a device other than the device that most recently commanded the object.

Test Concept: Command an object, O1, that supports the value source mechanism, from device D1, and verify the updated Value_Source. Attempt to write to the Value_Source property from device D2. Verify that an error is returned and Value_Source does not change.

Test Steps:

1. TRANSMIT WriteProperty-Request,
   SOURCE = D1,
   'Object Identifier' = O1,
   'Property Identifier' = (P1: the property monitored by the Value_Source mechanism for this object type),
   'Priority' = (PRIO: absent or any value other than 6)
   'Property Value' = (X2: any valid value)
2. RECEIVE BACnet-Simple-ACK-PDU
3. IF (O1 is commandable) THEN
   VERIFY Priority_Array = X2, ARRAY_INDEX = PRIO
   ELSE
   VERIFY (P1) = X2
4. VERIFY Value_Source = (D1's device identifier or network address)
5. TRANSMIT WriteProperty-Request,
   SOURCE = D2,
   'Object Identifier' = O1,
   'Property Identifier' = Value_Source,
   'Priority' = PRIO,
   'Property Value' = (any valid value)
6. RECEIVE BACnet-Error PDU,
   'Error Class' = PROPERTY,
   'Error Code' = WRITE_ACCESS_DENIED
7. VERIFY Value_Source = (D1's device identifier or network address)

7.3.1.X42.Y2 Non-commandable Value_Source Property Test

Purpose: To verify that the Value_Source property indicates the source of the current Present_Value in a non-commandable object.

Test Concept: Select a non-commandable object with a writable Present_Value which supports the Value Source mechanism. Present_Value is written, and it is verified that Value_Source is updated appropriately. Value_Source is then written to verify that the last writer can update it.

Test Steps:

1. WRITE Present_Value = V1
2. VERIFY Present_Value = V1
3. VERIFY Value_Source = (TD's device identifier or network address)
4. WRITE Value_Source = (any valid value, V2)
5. VERIFY Value_Source = V2

7.3.1.X42.Y3 Value_Source Property None Test

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

Purpose: To verify that the Value_Source property shall have the value 'None' when there is no active value source.

Test Concept: If there is no active value source, i.e. the Present_Value has taken on the value of Relinquish_Default, then the Value_Source property shall have the value 'None'.

Configuration Requirements: The object (O1) to be tested shall have 1 non-NULL entry in its Priority_Array and the Current_Command_Priority has a value other than NULL or 6.

Test Steps:

1. READ PRIO = Current_Command_Priority
2. CHECK(PRIO <> 6 and PRIO <> NULL)
3. VERIFY Value_Source = (is not 'None')
4. WRITE Present_Value = NULL, PRIORITY = PRIO
5. VERIFY Last_Command_Time ~= (the current local time)
6. IF (O1 has Minimum_On_Time or Minimum_Off_Time properties) THEN
   WAIT the larger of Minimum_Off_Time and Minimum_On_Time
7. VERIFY Current_Command_Priority = NULL
8. VERIFY Value_Source = 'None' -- the value is the choice 'none'

7.3.1.X42.Y4 Commandable Value Source Test

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

Purpose: To verify that the Value_Source, Value_Source_Array, and Last_Command_Time update correctly when Present_Value is written in a commandable object.

Test Concept: A commandable object which supports the value source mechanism is selected for the test. The Present_Value is written. Last_Command_Time, Value_Source and Value_Source_Array properties are checked to verify that they have been updated appropriately. Value_Source is then written, and it is verified that Last_Command_Time property has not changed.

Configuration Requirements: The object being tested shall be commandable and support the Value Source mechanism. No other internal processes shall be controlling the object.

Test Steps:
-- Verify that the value source properties are updated when Present_Value is commanded.

1. WRITE Present_Value = (V1: any valid value), PRIORITY = (PRIO: any value other than 6)
2. VERIFY Value_Source_Array = (SRC1: TD's device identifier or network address),
3. ARRAY_INDEX = PRIO
4. VERIFY Value_Source = SRC1
5. VERIFY Last_Command_Time ~= (the current local time)

-- Verify that Value_Source can be written and that Last_Command_Time does not update.

6. READ T1 = (O1), Last_Command_Time
7. WRITE Value_Source = (SRC2: any valid value), PRIORITY = PRIO
8. VERIFY Value_Source_Array = SRC2, ARRAY_INDEX = PRIO
9. IF (Current_Command_Priority == PRIO) THEN
   VERIFY Value_Source = SRC2
10. VERIFY Last_Command_Time = T1

7.3.1.X42.Y5 Life Safety Value_Source Property Test

Purpose: To verify that the Value_Source property indicates the source of the current Mode property in a life safety object.

Test Concept: Select a life safety object which supports the Value Source mechanism. Mode is written, and it is verified that Value_Source is updated appropriately. Value_Source is then written to verify that the last writer can update it.

Test Steps:

1. WRITE Mode = V1
2. VERIFY Mode = V1
3. VERIFY Value_Source = (TD's device identifier or network address)
4. WRITE Value_Source = (any valid value, V2)
5. VERIFY Value_Source = V2

7.3.1.6.11 Minimum_Off_Time - Value Source Mechanism

Purpose: To verify that the value source used for priority 6 is the commanded object while Minimum_Off_Time is in effect.

Test Concept: A commandable object which supports the value source mechanism is selected for the test. When Minimum_Off_Time takes effect, the Present_Value is written. The Value_Source and Value_Source_Array properties are monitored to verify that the source for priority 6 is the commanded object.

Configuration Requirements: The object, O1, to be tested shall be configured such that slot 6 in the Priority_Array and Value_Source_Array has a value of NULL. The object being tested must also be configured with Minimum_Off_Time values sufficiently large enough to allow execution of this test. If no object exists with Minimum_Off_Time property, this test shall be skipped.

Test Steps:
1. VERIFY Value_Source = (any valid value)
2. VERIFY Priority_ Array = NULL, ARRAY INDEX = 6
3. VERIFY Value_Source_Array = NULL, ARRAY INDEX = 6
4. WRITE Present_Value = INACTIVE, PRIORITY > 6
5. VERIFY Present_Value = INACTIVE
6. VERIFY Priority_Array = INACTIVE, ARRAY INDEX = 6
7. VERIFY Value_Source = O1
8. VERIFY Value_Source_Array = O1, ARRAY INDEX = 6
9. WAIT (**Minimum ON/OFF Fail Time** + Minimum_Off_Time)
10. VERIFY Value_Source = 'None'

7.3.1.6.12 Minimum_On_Time - Value Source Mechanism

Purpose: To verify that the value source used for priority 6 is the commanded object while Minimum_On_Time is in effect.

Test Concept: A commandable object which supports the value source mechanism is selected for the test. When Minimum_On_Time takes effect, the Present_Value is written. The Value_Source and Value_Source_Array properties are monitored to verify that the source for priority 6 is the commanded object.

Configuration Requirements: The object, O1, to be tested shall be configured such that slot 6 in the Priority_Array and Value_Source_Array has a value of NULL. The object being tested must also be configured with Minimum_On_Time values sufficiently large enough to allow execution of this test. If no object exists with Minimum_On_Time property, this test shall be skipped.

Test Steps:
1. VERIFY Value_Source = (any valid value)
2. VERIFY Priority_ Array = NULL, ARRAY INDEX = 6
3. VERIFY Value_Source_Array = NULL, ARRAY INDEX = 6
4. WRITE Present_Value = ACTIVE, PRIORITY > 6
5. VERIFY Present_Value = ACTIVE
6. VERIFY Priority_Array = ACTIVE, ARRAY INDEX = 6
7. VERIFY Value_Source = O1
8. VERIFY Value_Source_Array = O1, ARRAY INDEX = 6
9. WAIT (**Minimum ON/OFF Fail Time** + Minimum_On_Time)
10. VERIFY Value_Source = 'None'

[The tests for COVM were removed because they are included in addenda aq]