BACnet® TESTING LABORATORIES
ADDENDA

Addendum r to BTL Test Package 15

Revision 6
Revised 8/6/2019

Approved by the BTL Working Group on June 20, 2019;
Approved by the BTL Working Group Voting Members on August 6, 2019;
Published on August 7, 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 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-15.2r-1: Update Test Plan Directions for AE-VM-A and AE-M-A [BTLWG-294]

Overview:

The supported services and algorithms for AE-VM-A are a subset of those for AE-AVM-A, so the tests for any devices claiming support for both will satisfy all requirements for AE-VM-A by satisfying the requirements for AE-AVM-A.

Changes:

[In BTL Test Plan, update Test Conditionality, Test Directives and Testing Hints as shown below for AE-VM-A]

5.17 Alarm and Event Management - View Modify - A

5.17.1 Base Requirements

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

135.1-2013 - 8.18.3 - Reading and Presenting Properties		
	Test Conditionality	Must be executed. *Note: If the IUT also claims support for AE-AVM-A, this test may be omitted.*
	Test Directives	*Repeat the test for each of the following event generating objects: Analog Input/Output/Value, Accumulator, Event Enrollment, Loop, and Pulse Converter.* *Repeat for each of the required properties listed in the table in the BIBB definition.*
	Testing Hints	~~Repeat the test for each of the following event generating objects: Analog Input/Output/Value, Accumulator, Event Enrollment, Loop, and Pulse Converter.~~ ~~Repeat for each of the required properties listed in the table in the BIBB definition.~~

135.1-2013 - 8.22.4 - Accepting Input and Modifying Properties		
	Test Conditionality	Must be executed. *Note: If the IUT claims support for AE-AVM-A, this test may be omitted.*
	Test Directives	*Repeat the test for each of the following event generating objects: Analog Input/Output/Value, Accumulator, Event Enrollment, Loop, and Pulse Converter.* *Repeat for each of the required properties listed in the table in the BIBB definition.* *Repeat the test for a variety of values that cover the range of values required by the BIBB.*
	Testing Hints	~~Repeat the test for each of the following event generating objects: Analog Input/Output/Value, Accumulator, Event Enrollment, Loop, and Pulse Converter.~~ ~~Repeat for each of the required properties listed in the table in the BIBB definition.~~ ~~Repeat the test for a variety of values that cover the range of values required by the BIBB.~~

[In BTL Test Plan, update Test Conditionality, Test Directives and Testing Hints as shown below for AE-AVM-A]

5.19 Alarm and Event Management - Advanced View Modify - A

5.19.1 Base Requirements

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

135.1-2013 - 8.18.3 - Reading and Presenting Properties		
	Test Conditionality	Must be executed.
	Test Directives	*Repeat the test for each of the following event generating objects: Analog Input/Output/Value, Accumulator, Event Enrollment, Loop, and Pulse Converter.*

		Repeat for each of the required properties listed in the table in the BIBB definition.
	Testing Hints	~~Repeat the test for each of the following event generating objects: Analog Input/Output/Value, Accumulator, Event Enrollment, Loop, and Pulse Converter.~~ ~~Repeat for each of the required properties listed in the table in the BIBB definition.~~
135.1-2013 - 8.22.4 - Accepting Input and Modifying Properties		
	Test Conditionality	Must be executed.
	Test Directives	*Repeat the test for each of the following event generating objects: Analog Input/Output/Value, Accumulator, Event Enrollment, Loop, and Pulse Converter.* *Repeat for each of the required properties listed in the table in the BIBB definition.* *Repeat the test for a variety of values that cover the range of values required by the BIBB.*
	Testing Hints	~~Repeat the test for each of the following event generating objects: Analog Input/Output/Value, Accumulator, Event Enrollment, Loop, and Pulse Converter.~~ ~~Repeat for each of the required properties listed in the table in the BIBB definition.~~ ~~Repeat the test for a variety of values that cover the range of values required by the BIBB.~~

BTL-15.2r-2: Unsupported Message Text Directive Changes [BTLWG-326]

Overview:

BTL CR-0402 identified that the Test Directives for 9.1.X1 would result in too many executions of the test. The Test Directives were removed match the similar test handling for EventNotifications (i.e. 9.5.X1 and 9.4.X1).

Changes:

[In BTL Test Plan, modify Test Directives for test 9.1.X1 located in Section AE-ACK-B Base Requirements (5.5.1)]

BTL - 9.1.X1 - Unsupported Message Text Character Set AcknowledgeAlarm Test		
	Test Conditionality	If the IUT supports all character sets, this test shall be skipped.
	Test Directives	~~The objects selected by the tester should include all variants that differ in the set of supported intrinsic alarming properties, or the writability of any of those properties. At least one instance of each variant should be selected. Repeat for all supported transitions, if different transition types are supported in any object, as each may have different behavior.~~
	Testing Hints	

BTL-15.2r-3: Minimum On/Off Test Changes [BTLWG-399]

Overview:

BTL-CR-0433 pointed out a problem with the minimum on/off testing in 7.3.1.6.10 and the BTL-WG decided that the notes to tester should be changed.

Changes:

[In BTL Test Plan, change the name of test 7.3.1.6.10, to correct the misspelled verb in its name.]

[In BTL Specified Tests, change test 7.3.1.6.10, and correct the misspelled verb in its name and update Notes to Tester.]

7.3.1.6.10 Ensuring Minimum Times Are Not ~~E~~*A*ffected By Time Changes

Reason for Change: This test is not specified in any SSPC proposal.

Purpose: To verify that minimum times are not affected by changing the time in a device via TimeSynchronization or UTCTimeSynchronization requests.

Test Concept: The initial Present_Value of the object being tested is set to ACTIVE and the value at slot 6 in the priority-array has a value of NULL. Present_Value of the object is written to INACTIVE such that present-value and slot 6 in the priority-array change to INACTIVE. Before Minimum_Off_Time expires, the time is changed to a value T1 which is more than Minimum_Off_Time in the future and Present_Value and Slot 6 in the priority-array are read to verify that they were not affected by the time change. After Minimum_Off_Time expires, slot 6 in the priority-array is read again to verify that it is no longer INACTIVE.

Configuration Requirements: The object to be tested shall be configured such that the present value is set to ACTIVE and slot 6 in the Priority_Array has a value of NULL. If the IUT does not support TimeSynchronization or UTC-TimeSynchronization, then this test shall be omitted.

Test Steps:

1. VERIFY Present_Value = ACTIVE
2. VERIFY Priority_ Array = NULL, ARRAY_INDEX = 6
3. WRITE Present_Value = INACTIVE
4. VERIFY Present_Value = INACTIVE
5. VERIFY Priority_Array = INACTIVE, ARRAY_INDEX = 6
6. TRANSMIT
    DA = GLOBAL BROADCAST,
    SA = TD
    BACnet-Unconfirmed-Request-PDU,
        'Service Choice' =   TimeSynchronization-Request,
        Date =   T1,
        Time =   T1
7. TRANSMIT
    DA = GLOBAL BROADCAST,
    SA = TD
    BACnet-Unconfirmed-Request-PDU,
        'Service Choice' =   UTC-TimeSynchronization-Request,
        Date =   T1,
        Time =   T1
8. VERIFY Priority_Array = INACTIVE, ARRAY_INDEX = 6
9. WAIT (the remainder of Minimum_Off_Time)
10. VERIFY Priority_Array <> INACTIVE, ARRAY_INDEX = 6

Notes to Tester: The test above is written for Minimum_Off_Time. To execute this test for Minimum_On_Time, use INACTIVE where ACTIVE is specified, ACTIVE where INACTIVE is specified, and Minimum_On_Time where Minimum_Off_Time is specified.

BTL-15.2r-4: Clarify Requirements for Segmentation Tests [BTLWG-425]

Overview:

Clause 12.11.20 was changed based on interpretation requests IC135-2010-11 and IC135-2012-12 with the addition of the following sentences:

> If the Segmentation_Supported property of the Device object has a value of SEGMENTED_TRANSMIT or SEGMENTED_NONE, this property shall have a value of 1. If the Segmentation_Supported property of the Device object has a value of SEGMENTED_BOTH or SEGMENTED_RECEIVE, then the value of this property shall have a value greater than 1.

The Segmentation_Supported property test is changed to test for these new cases.

Changes:

[In BTL Specified Tests, Update purpose and Test Steps as shown]

7.3.2.10.X2 Max_Segments_Accepted at least the minimum

Reason for Change: There is no SSPC proposal for this change.

Purpose: To verify that the IUT correctly implements the Max_Segments_Accepted property value when it does support segmentation.

Configuration Requirements: If the IUT cannot be configured to support segmentation, then this test shall be skipped.

Test Steps:

```
1.  IF Segmentation_Supported == SEGMENTED_TRANSMIT or
        Segmentation_Supported == SEGMENTED_NONE
            VERIFY Max_Segments_Accepted = 1
    ELSE
            VERIFY (Max_Segments_Accepted > 1)
```

BTL-15.2r-5: Writing Non-Array Properties with an Array Index Test Fixes [BTLWG-432]

Overview:

Tests BTL - 9.23.2.4 and BTL - 9.23.2.5 are the negative test cases under Data Sharing - WritePropertyMultiple - B and are missing the Property Array Index parameter in the WritePropertyMultiple-Error

Changes:

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

9.23.2.4 Writing Non-Array Properties with an Array Index

Purpose: This test case verifies that the IUT can execute WritePropertyMultiple service requests when the property value is not an array, but an array index is included in the service request. This test shall only be performed if Protocol_Revision is present and has a value greater than or equal to 4.

Test Concept: The TD shall select an object, designated Object1, in the IUT that contains a writable scalar property designated P1 *having value X*. An attempt will be made to write to this property using an array index *Y*. If no suitable object can be found, then this test shall be omitted.

Configuration Requirements: If the IUT supports any writable properties that are scalars, it shall be configured with at least one such property that can be used for this test.

Test Steps:

```
1.  READ X = (Object1), P1
~~1.  VERIFY (Object1), P1 = (the value defined for this property in the EPICS)~~
2.  TRANSMIT WritePropertyMultiple-Request,
        'Object Identifier' = Object1,
        'Property Identifier' = P1,
        'Property Value' = (any valid value of the correct datatype for this property subject to the   restrictions
                                specified in the EPICS as defined in 4.4.2, except the value X read in step 1),
        'Property Array Index' = Y (any positive integer)
3.  RECEIVE WritePropertyMultiple-Error,
        'Error Class' = PROPERTY,
        'Error Code' = PROPERTY_IS_NOT_AN_ARRAY,
        'Object Identifier' = Object1,
        'Property Identifier' = P1
        'Property Array Index' = Y
4.  VERIFY (Object1), P1 = X ~~(the value defined for this property in the EPICS)~~
```

9.23.2.5 Writing Array Properties with an Array Index that is Out of Range

Purpose: This test case verifies that the IUT can execute WritePropertyMultiple service requests when the requested property value is an array, but the array index is out of range. This test shall only be performed if Protocol_Revision is present and has a value greater than or equal to 4.

Test Concept: The TD shall select an object, designated Object1, in the IUT that contains a writable array property designated P1 *having value X*. An attempt will be made to write to this property using an array index *Y* that is out of range. If no suitable object can be found, then this test shall be omitted.

Configuration Requirements: If the IUT supports any writable properties that are arrays, it shall be configured with at least one such property t

Test Steps:

1. READ X = (Object1), P1

~~1. VERIFY (Object1), P1 = (the value defined for this property in the EPICS)~~

2. TRANSMIT WritePropertyMultiple-Request,

    'Object Identifier' = Object1,

    'Property Identifier' = P1,

    'Property Value' = (any *valid* value of the correct datatype for this property subject to the restrictions specified in the EPICS as defined in 4.4.2, except the value *X* read in step 1),

    'Property Array Index' = *Y* (any ~~positive integer~~ *value* that is larger *than*~~that~~ the *current* ~~supported~~ size of the array)

3. RECEIVE WritePropertyMultiple-Error,

    'Error Class' = PROPERTY,

    'Error Code' = INVALID_ARRAY_INDEX,

    'Object Identifier' = Object1,

    'Property Identifier' = P1

    'Property Array Index' = Y

4. VERIFY (Object1), P1 = *X* ~~(the value defined for this property in the EPICS)~~

BTL-15.2r-6: Added Missing Test to AE-EL-I-B Test Plan [BTLWG-507]

Overview:
Test 135.1-2013 - 7.3.2.24.6.1 - Stop_When_Full TRUE Test is referenced under Alarms and Event Management - Event Log - External - B, however same is missing under Alarms and Event Management - Event Log –Internal - B. Added the testcase under Alarms and Event Management - Event Log -Internal - B in the Test Plan.

Changes:

[In Test Plan sections 5.14.1 add 135.1-2013 - 7.3.2.24.6.1]

5.14 Alarm and Event Management - Event Log - Internal - B
5.14.1 Base Requirements

…		
135.1-2013 - 7.3.2.24.6.1 - Stop_When_Full TRUE Test		
	Test Conditionality	*This shall be executed only if the property is configurable or equal to TRUE*
	Test Directives	
	Testing Hints	
…		

BTL-15.2r-7: Update Test Plan Directives for DS-V-A, DS-AV-A, DS-M-A, and DS-AM-A [BTLWG-539]

Overview:

Testing Hints were removed and placed in the appropriate Test Conditionality and Test Directives locations.

Changes:

[In BTL Test Plan, for DS-V-A, update the Test Directives and Test Conditionality as shown]

4.11 Data Sharing - View - A
4.11.1 Base Requirements
Base requirements must be met by any IUT claiming conformance to this BIBB.

135.1-2013 - 8.18.3 - Reading and Presenting Properties		
	Test Conditionality	Must be executed. *Note: if the IUT also claims support for DS-AV-A, this test may be omitted.*
	Test Directives	*Repeat the test for <u>each</u> of the standard object types and associated properties specified by DS-V-A.*
	Testing Hints	~~Repeat the test for each of the standard object types and associated properties specified by DS-V-A.~~ ~~Note: If the IUT also supports DS-AV-A, this test should be omitted because it is a subset of the DS-V-A test~~.

[In BTL Test Plan, for DS-AV-A, update the Test Directives and Test Conditionality as shown]

4.12 Data Sharing - Advanced View - A
4.12.1 Base Requirements
Base requirements must be met by any IUT claiming conformance to this BIBB.

135.1-2013 - 8.18.3 - Reading and Presenting Properties		
	Test Conditionality	Must be executed.
	Test Directives	*Repeat the test for <u>all</u> standard objects and properties, excluding the Life Safety and Access Control objects, and the Object_Identifier and Object_Type properties.*
	Testing Hints	~~Repeat the test for all standard objects and properties, excluding the Life Safety and Access Control objects, and the Object_Identifier and Object_Type properties.~~ For properties that contain a CHOICE construct, the IUT shall be capable of reading and presenting each of the forms of the datatype as defined in the IUT's claimed protocol revision. Full accuracy presentation is not required throughout the IUT, but there should be at least one place provided by the IUT that allows the presentation of each property to be presented in such a way that the presentation requirements of DS-AV-A are met. ~~Note: If this test is applied, the tests for DS-V-A may be omitted.~~

[In BTL Test Plan, for DS-M-A, update the Test Directives and Test Conditionality as shown]

4.13 Data Sharing - Modify - A
4.13.1 Base Requirements
Base requirements must be met by any IUT claiming conformance to this BIBB.

135.1-2013 - 8.22.4 - Accepting Input and Modifying Properties		
	Test Conditionality	Must be executed*, unless the IUT also claims support for DS-AM-A*.

	Test Directives	*Repeat the test for each of the required object types listed in the table in the BIBB definition.* *Repeat for each of the required properties listed in the table in the BIBB definition, except for those properties which are commandable.* *Repeat the test for a variety of values that cover the range of values required by the "Minimum Writable Value Ranges" table in the DS-M-A BIBB definition.*
	Testing Hints	~~Repeat the test for each of the required object types listed in the table in the BIBB definition.~~ ~~Repeat for each of the required properties listed in the table in the BIBB definition, except for those properties which are commandable.~~ ~~Repeat the test for a variety of values that cover the range of values required by the "Minimum Writable Value Ranges" table in the DS-M-A BIBB definition.~~ ~~Note: If the IUT also supports DS-AM-A, this test may be omitted because it is a subset of the DS-AM-A test.~~
135.1-2013 - 8.22.5 - Accepting Input and Commanding/Relinquishing Properties		
	Test Conditionality	Must be executed*, unless the IUT also claims support for DS-AM-A*.
	Test Directives	*This test should be executed at priority 8 only, i.e.* $PR_1 = 8$.
	Testing Hints	~~This test should be executed at priority 8 only, i.e. PR~~$_1$ ~~= 8.~~ ~~Note: If the IUT also supports DS-AM-A, this test may be omitted because it is a subset of the DS-AM-A test.~~

[In BTL Test Plan, for DS-AM-A, update the Test Directives and Test Conditionality as shown]

4.14 Data Sharing - Advanced Modify - A

4.14.1 Base Requirements

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

135.1-2013 - 8.22.4 - Accepting Input and Modifying Properties		
	Test Conditionality	Must be executed.
	Test Directives	*Repeat the test for all standard objects and properties, excluding the Life Safety and Access Control objects, and the Object_Identifier and Object_Type properties. Also exclude any properties that are required to be read-only by the BACnet standard, and exclude properties which are commandable because those are covered by a different test.* *Repeat the test for a variety of values that cover the range of values required by the "Minimum Writable Value Ranges" table in the DS-M-A BIBB definition.*
	Testing Hints	~~Repeat the test for all standard objects and properties, excluding the Life Safety and Access Control objects, and the Object_Identifier and Object_Type properties. Also exclude any properties that are required to be read-only by the BACnet standard, and exclude properties which are commandable because those are covered by a different test.~~ ~~Repeat the test for a variety of values that cover the range of values required by the "Minimum Writable Value Ranges" table in the DS-M-A BIBB definition.~~ ~~Note: If this test is applied, the tests for DS-M-A may be omitted.~~
135.1-2013 - 8.22.5 - Accepting Input and Commanding/Relinquishing Properties		
	Test Conditionality	Must be executed.
	Test Directives	*Repeat the test for all priority values, 1 through 16.*
	Testing Hints	~~Repeat the test for all priority values, 1 through 16.~~

BTL-15.2r-8: Accumulator Out_Of_Service Conditionality [BTLWG-652]

Overview:

In step 5, the test currently requires that the IN_ALARM be TRUE. This does not work for Accumulator objects which do not support event reporting.

Changes:

[In BTL Specified Tests, Modify 7.3.2.X37.1.6 Out_Of_Service Accumulator Test in BTL Specified Tests]

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 = (?, FALSE, ?, 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.  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)
```