BACnet® TESTING LABORATORIES
ADDENDA

Addendum misc3 to BTL Test Package 16.1

Revision 3
Revised September 30, 2020

Approved by the BTL Working Group on June 4, 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-16.1misc3-1:LifeSafetyOperation Service Execution Tests - BTLWG-889

Overview:

The existing LifeSafetyOperation tests are improved both in language and approach

Changes:

Test Plan Changes

[Modify in AE-LS-B, section 5.21.1]

5.22.1 Base Requirements

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

…		
~~**135.1-2013 - 9.9.1 - Reset Single Object Execution Tests**~~ ***BTL - 9.9.1.1 - Reset Single Object Execution Tests***		
	Test Conditionality	~~Must be executed.~~*This test shall be skipped if the IUT does not support a life safety object which latches.*
	Test Directives	
	Testing Hints	
~~**135.1-2013 - 9.9.2 - Reset Multiple Object Execution Test**~~ ***BTL - 9.9.1.2 - Reset Multiple Object Execution Test***		
	Test Conditionality	~~Must be executed.~~*This test shall be skipped if the IUT does not support a life safety object which latches.*
	Test Directives	
	Testing Hints	
~~**135.1-2013 - 9.9.3 - Silencing Execution Test**~~ ***BTL - 9.9.1.3 - Silencing Execution Test***		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 9.9.2.1 LifeSafetyOperation for an Object Which Does Not Exist		
	Test Conditionality	*Must be executed.*
	Test Directives	
	Testing Hints	
BTL - 9.9.2.2 LifeSafetyOperation which is Invalid given the Object's Current State		
	Test Conditionality	*If there is no life safety object which will reject a lifesafety operation because of its current state, this test shall be skipped.*
	Test Directives	
	Testing Hints	
BTL - 9.9.2.3 LifeSafetyOperation On An Object Which Does Not Support It		
	Test Conditionality	*If the IUT claims Protocol_Revision 20 or lower, this test shall be skipped.* *If there is no object in the IUT which does not support lifesafety operations, this test shall be skipped.*
	Test Directives	
	Testing Hints	

BTL Specified Tests Changes

[In BTL Specified Tests, move existing tests into BTL Specified Tests and renumber and modify by adding new section 9.9.1 and putting the tests under the new section]

9.9 LifeSafetyOperation Service Execution Test

9.9.1 Positive LifeSafetyOperation Execution Tests

~~9.9.1~~*9.9.1.1* Reset Single Object Execution Tests

Reason for Change: The test is written to not rely on Event_State so it can be applied to objects which do not support event reporting.

~~BACnet Reference Clause: 13.13.~~

Purpose: To verify that the IUT *correctly resets a latched life safety object when executing* ~~can correctly execute~~ a LifeSafetyOperation service request ~~to a single~~ *directed at the* ~~Life Safety Object~~ *life safety object*.

Test Concept: A ~~Life Safety~~ *life safety* object*, which latches it's Present_Value,* is toggled between a ~~NORMAL and OFFNORMAL state~~ *normal and non-normal BACnetLifeSafetyState*. The ~~STATUS~~ *object's Present_Value* should latch in the ~~OFFNORMAL~~ *non-normal* state until the LifeSafetyOperation service request is ~~transmitted~~ *executed. This is repeated for each LifeSafetyOperation request type supported by the device.* ~~transmitted. This test may be omitted when the device does not support latching.~~

Configuration Requirements: ~~The IUT must support a Life Safety Object that supports alarming. This object is configured into the NORMAL state, driven to the OFFNORMAL state and returned to the NORMAL state by adjusting its Present_Value. The STATUS should remain latched in the OFFNORMAL or FAULT state until the reset is issued. This test must be repeated for each reset function supported.~~ *The Life-Safety object starts the test with its Present_Value in a normal state as interpreted by the life safety object. This test shall be skipped if the device does not support latching.*

Test Steps:

```
1.  REPEAT X = (All supported enumerations that reset the object) DO {
2.      MAKE (the selected object ~~enter~~ *enters* a latched *non-normal* state where enumeration X will reset the object)
*3.     VERIFY Present_Value = (S1: a non-NORMAL state)*
*4.     VERIFY Tracking_Value = S1*
5.      MAKE (~~Event_State = NORMAL~~*remove the non-normal condition*)
6.      *VERIFY Present_Value = S1*
*7.     VERIFY Tracking_Value = (S2: a NORMAL state)*
~~        IF (enumeration value creates an Event_State = FAULT) THEN~~
~~            CHECK (Event_State = FAULT)~~
~~        ELSE~~
~~            CHECK (Event_State = OFFNORMAL)~~
~~6~~*7*.   TRANSMIT LifeSafetyOperation-Request,
            'Requesting Process Identifier' =   (any valid identifier),
            'Requesting Source' =               (any valid character string),
            'Request' =                         *X,* ~~(any valid LifeSafetyOperation request),~~
            'Object Identifier' =               (the selected object)
~~7~~*8*.   RECEIVE BACnet-SimpleACK-PDU
~~8~~*8*.   VERIFY ~~(Object), STATUS~~ *Present_Value = S2* ~~NORMAL)~~
    }
```

~~9.9.2~~*9.9.1.2* Reset Multiple Object Execution Tests

Reason for Change: The test is generalized to work for any number of objects. The test is written to not rely on Event_State so it can be applied to objects which do not support event reporting.

~~BACnet Reference Clause: 13.13.~~

Purpose: To verify that the IUT *correctly resets multiple latched life safety objects when executing* ~~can correctly execute~~ a LifeSafetyOperation service request ~~to multiple Life Safety objects~~ *not directed at a specified life safety object*.

Test Concept: ~~Two~~ *Multiple* Life Safety objects, O1 … On, are toggled between a ~~NORMAL and an OFFNORMAL state~~ *normal and non-normal BACnetLifeSafetyState*. The ~~STATUS~~ *objects' Present_Value properties* should latch in the ~~OFFNORMAL~~ *non-normal* state until the LifeSafetyOperation service request is ~~transmitted~~ *executed*. *This is repeated for each LifeSafetyOperation request type supported by the device.* ~~This test may be omitted when the device does not support latching.~~

Configuration Requirements: ~~The IUT must support a Life Safety Object that supports alarming and have at least two objects that can be latched. This object is configured into the NORMAL state, driven to the OFFNORMAL state~~ ~~*or FAULT*~~ ~~and returned to the NORMAL state by adjusting its Present_Value. The STATUS should remain latched in the OFFNORMAL state until the reset is issued. This test must be repeated for each reset function supported.~~ *The Life-Safety objects start the test with their Present_Value properties in a normal state as interpreted by the life safety objects. This test shall be skipped if the device does not support latching Life Safety objects. If the IUT supports a single latching life safety object, apply this test to the single object.*

Test Steps:

```
1.  REPEAT X = (All supported enumerations that reset the *objects* ~~object~~) DO {
2.      MAKE (the selected objects enter ~~a~~ latched *non-normal* ~~state~~ *states* where enumeration X will reset the
                objects)
3.      *REPEAT Oi = (each selected life safety object) {*
            *VERIFY Oi, Present_Value = (Si: a non-normal value)*
            *VERIFY Oi, Tracking_Value = Si*
        *}*
4.      MAKE (~~Event State = NORMAL~~*remove the non-normal conditions for both objects*)
5.      *REPEAT Oi = (each selected life safety object) {*
            *VERIFY Oi, Present_Value = Si*
            *VERIFY Oi, Tracking_Value = TVi*
        *}*
~~        IF (enumeration value creates an Event_State = FAULT) THEN~~
~~            CHECK (Event_State = FAULT)~~
~~        ELSE~~
~~            CHECK (Event_State = OFFNORMAL)~~
6.      TRANSMIT LifeSafetyOperation-Request,
            'Requesting Process Identifier' =   (any valid identifier),
            'Requesting Source' =               (any valid character string),
            'Request' =                         *X,* ~~(any valid LifeSafetyOperation request),~~
7.      RECEIVE BACnet-SimpleACK-PDU
~~7.      VERIFY (Object), STATUS = NORMAL~~
*8.*    *REPEAT Oi = (each selected life safety object) {*
            *VERIFY Oi, Present_Value = TVi*
        *}*
}
```

~~9.9.3~~*9.9.1.3* Silencing*/Unsilencing* Execution Tests

~~BACnet Reference Clause: 13.13.~~

Purpose: To verify that the IUT can correctly execute a LifeSafetyOperation service request to silence *and unsilence* an alarming device.

Test Concept: An audible device and/or visual device is attached to the IUT and is sounding/flashing, *because a life safety object has entered a non-normal state and the property Silenced is UNSILENCED.* A LifeSafetyOperation service request is transmitted to silence the sounder/strobe. *Then, the life safety object remains in the non-normal state with Silenced equal to SILENCED. A LifeSafetyOperation service request is transmitted to unsilence the sounder/strobe (reactivate it) and it is verified that the object is unsilenced.*

There are different allowable BACnetSilencedState values based on the silence operation performed and the setup of the IUT. In the below tables, N/A marks an operation that is inappropriate for the test with the corresponding IUT setup.

Only Sounder Attached				
Silence Operation	***Allowable Silenced State***		***Unsilenced Operation***	***Allowable Silenced State***
SILENCE	*ALL_SILENCED, AUDIBLE_SILENCED, proprietary*		*UNSILENCE*	*UNSILENCED, proprietary*
SILENCE_AUDIBLE	*ALL_SILENCED, AUDIBLE_SILENCED, proprietary*		*UNSILENCE_AUDIBLE*	*UNSILENCED, proprietary*
SILENCE_VISUAL	*N/A*		*UNSILENCE_VISUAL*	*N/A*

Only Strobe Attached				
Silence Operation	***Allowable Silenced State***		***Unsilenced Operation***	***Allowable Silenced State***
SILENCE	*ALL_SILENCED, VISUAL_SILENCED, proprietary*		*UNSILENCE*	*UNSILENCED, proprietary*
SILENCE_AUDIBLE	*N/A*		*UNSILENCE_AUDIBLE*	*N/A*
SILENCE_VISUAL	*ALL_SILENCED, VISUAL_SILENCED, proprietary*		*UNSILENCE_VISUAL*	*UNSILENCED, proprietary*

Sounder And Strobe Attached				
Silence Operation	***Allowable Silenced State***		***Unsilenced Operation***	***Allowable Silenced State***
SILENCE	*ALL_SILENCED, proprietary*		*UNSILENCE*	*UNSILENCED, proprietary*
SILENCE_AUDIBLE	*AUDIBLE_SILENCED, proprietary*		*UNSILENCE_AUDIBLE (all silenced)*	*SILENCED_VISUAL, proprietary*
			UNSILENCE_AUDIBLE (audible silenced, visual active)	*UNSILENCED, proprietary*
			UNSILENCE_AUDIBLE (audible active, visual silenced)	*N/A*
SILENCE_VISUAL	*VISUAL_SILENCED, proprietary*		*UNSILENCE_VISUAL (all silenced)*	*SILENCED_AUDIBLE, proprietary*
			UNSILENCE_VISUAL (audible silenced, visual active)	*N/A*
			UNSILENCE_VISUAL (audible active, visual silenced)	*UNSILENCED, proprietary*

Configuration Requirements: The IUT must be fitted with needed audible and visual equipment.

Test Steps:

1. REPEAT X = (All supported enumerations that silence the object) DO {
2.   MAKE (the selected object enter a state where enumeration X will *silence the sounder/strobe*~~commence alerts~~)
3.   *VERIFY Silenced = (Unsilenced or a proprietary value with a similar semantic)*

~~5. MAKE (Event_State = NORMAL)~~

~~6~~*4*.   TRANSMIT LifeSafetyOperation-Request,
        'Requesting Process Identifier' =   (any valid identifier),

```
            'Requesting Source' =                (any valid character string),
            'Request' =                          *X,* ~~(any valid LifeSafetyOperation request),~~
            'Object Identifier' =                (*absent or* the selected object)
~~7~~*5*.    RECEIVE BACnet-SimpleACK-PDU
~~8~~*6*.    CHECK (that the sounder/strobe is inactive)
~~9~~*7*.    *VERIFY Silenced = (an allowable silenced state based on the IUT setup and operation request X)*
~~10~~*8*.   *TRANSMIT LifeSafetyOperation-Request,*
            *'Requesting Process Identifier' =   (any valid identifier),*
            *'Requesting Source' =               (any valid character string),*
            *'Request' =                         (any valid LifeSafetyOperation request),*
            *'Object Identifier' =               (the selected object)*
*9.*     *RECEIVE BACnet-SimpleACK-PDU*
*10.*    *CHECK (the sounder / strobe active again, as appropriate to the operation)*
*11.*    *VERIFY Silenced = (the appropriate state based on the operation and IUT condition)*
    }
```

Notes to Tester: Source object needs to get silence only for configured objects.

[In BTL Specified tests, add new section and all new tests]

9.9.2 Negative LifeSafetyOperation Execution Tests

9.9.2.1 LifeSafetyOperation for an Object Which Does Not Exist

Purpose: To verify that the IUT correctly responds when a LifeSafetyOperation targets a non-existent object.

Test Concept: Send a LifeSafetyOperation request to the IUT targeting an object that does not exist in the IUT.

Test Steps:

```
1.  TRANSMIT LifeSafetyOperation-Request,
        'Requesting Process Identifier' =    (any valid value),
        'Requesting Source' =                (any valid value),
        'Request' =                          (any valid LifeSafetyOperation request supported by the device),
        'Object Identifier' =                (any object not contained in the IUT's database)
2.  RECEIVE BACnet-Error PDU,
        'Error Class' = OBJECT,
        'Error Code' = UNKNOWN_OBJECT'
```

9.9.2.2 LifeSafetyOperation which is Invalid given the Object's Current State

Purpose: To verify that the IUT correctly responds when a LifeSafetyOperation's Request value is invalid given the object's current state.

Test Concept: Send a LifeSafetyOperation request, with a Request value that is not valid for the current state, to the IUT.

Configuration Requirements: The life safety object, O1, being tested is placed into a state where Request R, a valid LifeSafetyOperation value which the life safety object would accept in other states, is currently invalid. If there is no such state, request value pair that satisfies this requirement, this test shall be skipped.

Test Steps:

```
1.  TRANSMIT LifeSafetyOperation-Request,
        'Requesting Process Identifier' =    (any valid value),
        'Requesting Source' =                (any valid value),
        'Request' =                          (R: a request value which is invalid given the life safety object's
                                             current state),
        'Object Identifier' =                O1
2.  RECEIVE BACnet-Error PDU,
        'Error Class' = OBJECT,
        'Error Code' = INVALID_OPERATION_IN_THIS_STATE
```

9.9.2.3 LifeSafetyOperation On An Object Which Does Not Support It

Purpose: To verify that the IUT correctly responds when a LifeSafetyOperation's is received that targets an object that does not support it.

Test Concept: Send a LifeSafetyOperation request, with an Object Identifier referencing an object in the IUT which does not support life safety operations.

Test Steps:

1. TRANSMIT LifeSafetyOperation-Request,
    'Requesting Process Identifier' = (any valid value),
    'Requesting Source' = (any valid value),
    'Request' = (any valid value normally supported by the IUT),
    'Object Identifier' = (an object in the IUT which does not support life safety operations)
2. RECEIVE BACnet-Error PDU,
    'Error Class' = OBJECT,
    'Error Code' = OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED