BACnet® TESTING LABORATORIES ADDENDA

Addendum bl to BTL Test Package 20.0.1

Revision v5
Revised 10/21/2022

Approved by the BTL Working Group on 2022-08-25.
Approved by the BTL Working Group Voting Members on 2022-10-21.
Published on 2022-10-26.

[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 20.0.1 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-20.0.1 bl-1: Clarify Out_Of_Service [BTLWG-681, BTLWG-1152, BTLWG-1157, BTLWG-1158]

Overview:

Changes were made per 135-2016bl-3 Clarify Out_Of_Service.

The Out_Of_Service functionality is inconsistent across objects and is unclear with respect the changeability of the Reliability property (vs writability). The Out_Of_Service property for all objects is modified to be consistent in requirements and presentation.

Changes:

Checklist Changes

[Modify all the Object sections, to change the word 'writable' to 'configurable'. Leave the S or O as is for the object being changed.]

XXX Object		
...		
	S | O	Supports configurable Out_Of_Service property
...		

Test Plan Changes

[Modify all Object sections per the following except 3.45.4 Access Zone Object, 3.44.2 Access Point Object and 3.56.3 Network Port Object]

X.X Supports Configurable Out_Of_Service Property

The Out_Of_Service property in Analog Output objects contained in the IUT is either writable or can be modified by any other means.

BTL - 7.3.1.1.X1 - Out_Of_Service, Status_Flags, and Reliability Tests		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

[Modify the Access Zone Object section to replace the test used from 7.3.1.1.X1 vs the 7.3.1.1.X3]

3.45.3 Supports Configurable Out_Of_Service Property

The Out_Of_Service property in Access Zone objects contained in the IUT is either writable or can be modified by any other means.

BTL - 7.3.1.1.X3 - Out_Of_Service, Status_Flags, and Reliability test for Objects without Present_Value		
	Test Conditionality	Must be executed.
	Test Directives	The test shall be executed using an Access Zone object
	Testing Hints	

[Modify the Access Point Object section 3.44.2, replace test 7.3.1.1X1 by 7.3.1.1X3, add a new test 7.3.1.X4]

3.44.2 Supports Configurable Out_Of_Service Property

The Out_Of_Service property in Access Point objects contained in the IUT is either writable or can be modified by any other means.

BTL - 7.3.1.1.X3 - Out_Of_Service, Status_Flags, and Reliability test for Objects without Present_Value		
	Test Conditionality	Must be executed.
	Test Directives	The test shall be executed using an Access Point object
	Testing Hints	
BTL - 7.3.1.1.X4 - Out_Of_Service, Access_Event for Access Point Objects Test		
	Test Conditionality	Must be executed.
	Test Directives	The test shall be executed using an Access Point object
	Testing Hints	

[Modify the Network Port Object section to add a new test BTL - 7.3.1.1.X5 to replace BTL - 7.3.1.1.X3]

3.56.3 Supports ~~Writable~~ *Configurable* Out_Of_Service Property

The Out_Of_Service property in Network Port objects contained in the IUT is either writable or can be modified by any other means.

BTL - 7.3.1.1.X5 - Out_Of_Service, Status_Flags, Reliability and Command Property Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

Specified Test Changes

[Add a new test 7.3.1.X4 into BTL Specified Tests]

7.3.1.1.X4 Out_Of_Service, Access_Event for Access Point Objects Test
Reason for Change: There is no test for this functionality.

Purpose: This test verifies the interrelationship between the Out_Of_Service and Access_Event properties. If 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.

Test Concept: Write to and verify the interrelationship between the Out_Of_Service, and Access_Event properties

Configuration Requirements: The selected object is configured such that Property Access_Event is not OUT_OF_SERVICE before execution of this test.

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 Access_Event = OUT_OF_SERVICE
4. IF (Out_Of_Service is writable) THEN
   WRITE Out_Of_Service = FALSE
   ELSE
   MAKE (Out_Of_Service = FALSE)
5. VERIFY Out_Of_Service = FALSE
6. VERIFY Access_Event = OUT_OF_SERVICE_RELINQUISHED

[Add a new test into BTL Specified Tests]

7.3.1.1.X5 - Out_Of_Service, Status_Flags, Reliability and Command Property Test
Reason for Change: There is no test for this functionality. New test per 135-2016bl-3

Purpose: This test verifies the interrelationship between the Out_Of_Service, Status_Flags, Reliability and Command properties.

Test Concept: Write to and verify the interrelationship between the Out_Of_Service, Status_Flags, Reliability and Command properties.

Configuration Requirements: The selected object is configured such that its Out_Of_Service shall be set to FALSE.
If exist the Command property shall be set to IDLE and Reliability to NO_FAULT_DETECTED.

Notes to Tester: When applying this test to a Network Port object in a non-routing device, once the Network Port object is out of service, the only remaining part of the test that can be executed is the verification that no BACnet communications occur through that network port. The rest of the test shall be skipped.

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. 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)
   }
5. IF (Command is present) THEN
   REPEAT Y = (all values of the BACnetNetworkPortCommand enumeration except RESTART_PORT, DISCONNECT, and DISCARD_CHANGES) DO {
   TRANSMIT WriteProperty-Request
   'Object Identifier' = (the object being tested),
   'Property Identifier'= Command,
   'Property Value' = Y
   RECEIVE BACnet-Error-PDU,
   'Error Class' = PROPERTY,
   'Error Code' = VALUE_OUT_OF_RANGE

   }

6. CHECK (functionality that should stop or be disabled is. All communication of the protocol modeled by the object, through that port is disabled)
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)

[Modify test 7.3.2.30.6]
7.3.2.30.6 Out_Of_Service Property Test
Reason for Change: Fixed Status_Flags expected values. New test per 135-2016bl-3.

BACnet Reference Clauses: 12.51.7

Purpose: This test case verifies that event forwarding is not done while Out_Of_Service is TRUE.

Test Concept: Set up both Recipient_List and Subscribed_Recipient recipient entries with no filters specified and then send event notifications to the Notification Forwarder while the value of the Out_Of_Service property is TRUE. Subscribed_Recipients are configured as part of base setup 2 for Notification Forwarder object tests. Verify that forwarding of the event notifications is not performed.

~~*If 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*~~.

Configuration Requirements: *The selected object is configured such that its Out_Of_Service shall be set to FALSE and Reliability set to NO_FAULT_DETECTED.* Base setup 2 for Notification Forwarder object tests with TR lifetime sufficient for this test.

Test Steps:

1. MAKE (Recipient_List = { (all), --Valid Days
   (all), --From Time, To Time
   DEST_OBJ_ID2, --Recipient D2
   DEST_PROCESS_ID, --Process Identifier
   FALSE, --Issue Confirmed Notifications
   {T, T, T} --Transitions
   }) --One list element
2. MAKE (Out_Of_Service = TRUE)
3. VERIFY Out_Of_Service = TRUE
4. VERIFY Status_Flags = (~~?~~*FALSE*, FALSE, ~~?~~*FALSE*, TRUE)
5. TRANSMIT SOURCE = DS, UnconfirmedEventNotification-Request,
   'Process Identifier' = SRC_PROCESS_ID,
   'Initiating Device Identifier' = SRC_NOTIF_DEV,
   'Event Object Identifier' = SRC_NOTIF_OBJ,
   'Time Stamp' = (any valid time stamp),
   'Notification Class' = SRC_NOTIF_CLS,
   'Priority' = (any valid priority),
   'Event Type' = (any valid event type),
   'Message Text' = (optional, any valid message text),
   'Notify Type' = SRC_NOTIF_TYP,
   'AckRequired' = (any valid value), -- absent if 'Notify Type' is ACK_NOTIFICATION
   'From State' = (any valid From_State), -- absent if 'Notify Type' is ACK_NOTIFICATION
   'To State' = (any valid To_State),
   'Event Values' = (any valid event values) -- absent if 'Notify Type' is ACK_NOTIFICATION
6. WAIT **Notification Fail Time**
7. CHECK (the IUT did not transmit an event notification)
8. MAKE (Out_Of_Service = FALSE)
9. VERIFY Out_Of_Service = FALSE
10. VERIFY Status_Flags = (~~?~~*FALSE*,~~?~~*FALSE*, ~~?~~*FALSE*, FALSE)
11. TRANSMIT SOURCE = DS, UnconfirmedEventNotification-Request,
    'Process Identifier' = SRC_PROCESS_ID,
    'Initiating Device Identifier' = SRC_NOTIF_DEV,
    'Event Object Identifier' = SRC_NOTIF_OBJ,
    'Time Stamp' = (any valid time stamp),
    'Notification Class' = SRC_NOTIF_CLS,
    'Priority' = (any valid priority),
    'Event Type' = (any valid event type),
    'Message Text' = (optional, any valid message text),
    'Notify Type' = SRC_NOTIF_TYP,
    'AckRequired' = (any valid value), -- absent if 'Notify Type' is ACK_NOTIFICATION
    'From State' = (any valid From_State), -- absent if 'Notify Type' is ACK_NOTIFICATION

'To State' = (any valid To_State),
'Event Values' = (any valid event values) -- absent if 'Notify Type' is ACK_NOTIFICATION

12. BEFORE **Notification Fail Time** --The following can be in any order
RECEIVE DESTINATION = D1, UnconfirmedEventNotification-Request
RECEIVE DESTINATION = D2, UnconfirmedEventNotification-Request
13. *MAKE (Out_Of_Service = TRUE)*
14. *VERIFY Out_Of_Service = TRUE*
15. *VERIFY Status_Flags = (FALSE, FALSE, FALSE, TRUE)*
16. *IF (Reliability is 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)
}
17. *IF (Out_Of_Service is writable) THEN*
WRITE Out_Of_Service = FALSE
ELSE
MAKE (Out_Of_Service = FALSE)
18. *VERIFY Out_Of_Service = FALSE*
19. *VERIFY Status_Flags = (?, FALSE, ?, FALSE)*