BACnet® TESTING LABORATORIES ADDENDA

Addendum g to BTL Test Package 15.1

Revision 3.0
Revised 10/17/2018

Approved by the BTL Working Group on <date>;
Approved by the BTL Working Group Voting Members on <date>;
Published on <date>.

[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 15.1 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 addition, 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-15.1g-1: BTL-R code in BACnet/IP BBMD to R [BTLWG-86]

Overview:
135-2010ad made these requirements in the standard, so change BTL-R codes in BACnet/IP BBMD Checklist section to R.

Changes:

[In BTL Checklist, modify the BACnet/IP BBMD section from BTL-R to R as shown below.]

9 Data Link Layer

Support	Listing	Option
…		
BACnet/IP - Annex J - BBMD		
	R	Base Requirements
	BTL-R	Supports a BDT with at least four entries
	~~BTL-R~~*R*	Registration by a Foreign Device is supported
	~~BTL-R~~*R*	Supports 2-hop mode
	O	Supports 1-hop mode
	O	BBMD supports Network Address Translation
	O	Is able to register as a Foreign Device
	O	Is able to initiate Original-Broadcast-NPDU
…		

BTL-15.1g-2: Add Fixed Length Array Tests [BTLWG-112]

Overview:
The existing test plan do not have a test to check the behavior where the Array is limited in length and we try to resize it using WriteProperty and WritePropertyMultiple service.

Changes:

[In Test Plan, add to section 4.6.1 as shown]

4.6 Data Sharing-WriteProperty - B

4.6.1 Base Requirements

…		
BTL - 9.22.2.X2 - Resizing a writable fixed size array property		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If IUT does not contain a writable fixed size array property, then this test shall be skipped.
	Test Directives	
	Testing Hints	For example, Weekly_Schedule.
	Notes & Results	

[In Test Plan, add to section 4.8.1 as shown]

4.8 Data Sharing-WritePropertyMultiple - B

4.8.1 Base Requirements

The IUT contains or can be made to contain writable fixed length array property.

…		
BTL - 9.23.2.X2 - Resizing a writable fixed size array property using WritePropertyMultiple service		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests.***
	Test Conditionality	If IUT does not contain a writable fixed size array property, then this test shall be skipped.
	Test Directives	
	Testing Hints	For example, Weekly_Schedule.
	Notes & Results	

[In BTL Specified Tests, Add this new test, as shown]

9.22.2.X2 Resizing a writable fixed size array property

Purpose: This test case verifies that the IUT correctly responds to an attempt to resize a writable fixed size array property using WriteProperty service.

Test Concept: Select an object (O1) in the IUT that contains a writable array property of a fixed size. This property is designated P1. If no suitable object can be found, then this test shall be omitted.

Test Steps:

1. READ X = (O1), P1 ARRAY INDEX = 0
2. WRITE P1= (Entire Array with any valid value greater than Array Size X)
3. RECEIVE BACnet-Error-PDU,
    'Error Class' = PROPERTY,
    'Error Code' = INVALID_ARRAY_INDEX | VALUE_OUT_OF_RANGE
4. VERIFY (O1), P1= X, ARRAY INDEX = 0
5. WRITE P1= (Entire Array with any valid value less than Array Size X)
6. RECEIVE BACnet-Error PDU,
    'Error Class' = PROPERTY,
    'Error Code' = INVALID_ARRAY_INDEX | VALUE_OUT_OF_RANGE
7. VERIFY (O1), P1= X, ARRAY INDEX = 0
8. WRITE P1 = (any valid value greater than Array Size X), ARRAY INDEX=0
9. RECEIVE BACnet-Error PDU,
    'Error Class' = PROPERTY,
    'Error Code' = INVALID_ARRAY_INDEX | VALUE_OUT_OF_RANGE
10. VERIFY (O1), P1= X, ARRAY INDEX = 0,
11. WRITE P1 = (any valid value less than Array Size X), ARRAY INDEX=0
12. RECEIVE BACnet-Error PDU,
    'Error Class' = PROPERTY,
    'Error Code' = INVALID_ARRAY_INDEX | VALUE_OUT_OF_RANGE
13. VERIFY (O1), P1= X, ARRAY INDEX = 0

9.23.2.X2 Resizing a writable fixed size array property using WritePropertyMultiple service

Purpose: This test case verifies that the IUT correctly responds to an attempt to resize a writable fixed size array property using WritePropertyMultiple service.

Test Concept: Select an object(O1) in the IUT that contains a writable array property of a fixed size. This property is designated P1. If no suitable object can be found, then this test shall be omitted.

Test Steps:
1. READ X = (O1), P1, ARRAY INDEX = 0
2. TRANSMIT WritePropertyMultiple-Request,
    'Object Identifier' = O1,
    'Property Identifier' = P1,
    'Property Value' = (Entire Array with any valid value greater than Array Size X)
3. RECEIVE WritePropertyMultiple-Error,
    'Error Class' = PROPERTY,
    'Error Code' = INVALID_ARRAY_INDEX | VALUE_OUT_OF_RANGE,
    'ObjectIdentifier' = O1,
    'PropertyIdentifier' = P1
4. VERIFY (O1), P1= X, ARRAY INDEX = 0
5. TRANSMIT WritePropertyMultiple-Request,
    'Object Identifier' = O1,
    'Property Identifier' = P1,
    'Property Value' = (Entire Array with any valid value less than Array Size X)
6. RECEIVE WritePropertyMultiple-Error,
    'Error Class’ = PROPERTY,
    'Error Code’ = INVALID_ARRAY_INDEX | VALUE_OUT_OF_RANGE,
    'ObjectIdentifier' = O1,
    'PropertyIdentifier' = P1
7. VERIFY (O1), P1= X, ARRAY INDEX = 0
8. TRANSMIT WritePropertyMultiple-Request,
    'Object Identifier' = O1,
    'Property Identifier' = P1,
    'Property Value' = (any valid value greater than Array Size X),

'Property Array Index' = 0
9. RECEIVE WritePropertyMultiple-Error,
'Error Class' = PROPERTY,
'Error Code' = INVALID_ARRAY_INDEX | VALUE_OUT_OF_RANGE,
'ObjectIdentifier' = O1,
'PropertyIdentifier' = P1
'Property Array Index'=0
10. VERIFY (O1), P1= X, ARRAY INDEX = 0
11. TRANSMIT WritePropertyMultiple-Request,
'Object Identifier' = O1,
'Property Identifier' = P1,
'Property Value' = (any valid value less than Array Size X),
'Property Array Index' = 0
12. RECEIVE WritePropertyMultiple-Error,
'Error Class' = PROPERTY,
'Error Code' = INVALID_ARRAY_INDEX | VALUE_OUT_OF_RANGE,
'ObjectIdentifier' = O1,
'PropertyIdentifier' = P1
'Property Array Index'= 0
13. VERIFY (O1), P1= X, ARRAY INDEX = 0

BTL-15.1g-3: Modify Wording of Subscribe Test Directives [BTLWG-305]

Overview:
Tests 8.10.1 and 8.10.2 have word 'must' in Test Plan Configuration as 'This test must not be executed with a lifetime of 0.' The word 'must' is not the proper term, and the whole directive could be moved instead into Test Directives, rather than Configuration.

Changes:

[In BTL Test Plan change the following sections]

4.9.3 Can Subscribe for Confirmed Notifications

The IUT can subscribe for, receive, and process confirmed Change of Value notifications.

135.1-2013 - 8.10.1 - Confirmed Notifications Subscription		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***. ~~This test must not be executed with a lifetime of 0.~~
	Test Conditionality	Must be executed.
	Test Directives	*This test shall not be executed with a lifetime of 0.*
	Testing Hints	
	Notes & Results	

4.9.4 Can Subscribe for Unconfirmed Notifications

The IUT can subscribe for, receive, and process unconfirmed Change of Value notifications.

135.1-2013 - 8.10.2 - Unconfirmed Notifications Subscription		
	Test Method	Manual
	Configuration	As per ***ASHRAE 135.1-2013***. ~~This test must not be executed with a lifetime of 0.~~
	Test Conditionality	Must be executed.
	Test Directives	*This test shall not be executed with a lifetime of 0.*
	Testing Hints	
	Notes & Results	

BTL-15.1g-4: Accurate COVU_Period Testing [BTLWG-307]

Overview:
This document modifies 8.3.X1 to more accurately test COVU_Period timing.

Changes:

[In BTL Test Plan, change test 8.3.X1]

8.3.X1 COVU_Recipients Notifications

Reason for Change: This test does not exist in 135.1.

Purpose: To verify that the IUT initiates UnconfirmedCOVNotification service requests to each entry in its COVU_Recipients property based on COVU_Period.

Test Concept: The IUT contains a Global Group object, O1, that is configured to periodically send UnconfirmedCOVNotification using COVU_Period and COVU_Recipients. The TD checks for these notifications.

Configuration Requirements: COVU_Recipients property shall be non-empty and contain at least one device and one address based recipient. The COVU_Period shall be non-zero.

Test Steps:

1. REPEAT X = (each entry in the COVU_Recipients) DO {
   BEFORE COVU_Period + **Notification Fail Time**
   RECEIVE UnconfirmedCOVNotification-Request,
   DESTINATION = X,
   'Subscriber Process Identifier' = 0,
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = O1,
   'Time Remaining' = 0,
   'List of Values' = (Member_Status_Flags, Elements of Present_Value)
   IF (X is the first entry in the COVU_Recipients) THEN
   READ T1 = Local_Time
   }
2. REPEAT X = (each entry in the COVU_Recipients) DO {
   BEFORE COVU_Period + **Notification Fail Time**
   RECEIVE UnconfirmedCOVNotification-Request,
   DESTINATION = X,
   'Subscriber Process Identifier' = 0,
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = O1,
   'Time Remaining' = 0,
   'List of Values' = (Member_Status_Flags, Elements of Present_Value)
   IF (X is the first entry in the COVU_Recipients) THEN
   READ T2 = Local_Time
   }
3. CHECK (T2 - T1 ~= COVU_Period)

Note to tester: The test shall pass regardless of the order in which the IUT generates the UnconfirmedCOVNotification-Requests in each step.

BTL-15.1g-5: Update Checklist to Protocol_Revision 20 [BTLWG-418]

Overview:
The checklist needs to be updated to contain all objects and BIBBs added for Protocol_Revision 20.

Includes addenda: bd, be, bi, bk, bl, bm, bn, bp, bq

Changes:

[In BTL Checklist, add these objects to end of Objects Section]

Support	Listing	Option
Staging Object		
	R[1,2]	Base Requirements
[1]Contact BTL for interim tests for this object.		
[2]Protocol_Revision 20 or higher must be claimed.		
Audit Reporter Object		
	R[1,2]	Base Requirements
[1]Contact BTL for interim tests for this object.		
[2]Protocol_Revision 20 or higher must be claimed.		
Audit Log Object		
	R[1,2]	Base Requirements
[1]Contact BTL for interim tests for this object.		
[2]Protocol_Revision 20 or higher must be claimed.		

[In BTL Checklist, modify the following BIBBs to include the new sections. They should be added directly below the last Conditional entry and before the Optional entry.]

Data Sharing - COV - A		
…		
	C[2,3,5]	Can subscribe for COV from Staging objects
. . .		
[3]Contact BTL for interim tests for this object. [5]Protocol_Revision 20 or higher must be claimed.		
Data Sharing - COV - B		
…		
	C[1,2,4]	Supports COV for Staging objects
…		
[2]Contact BTL for interim tests for this object. [4]Protocol_Revision 20 or higher must be claimed.		

[In BTL Checklist, add the following BIBBs to the end of the Data Sharing Section]

Data Sharing - Lighting Output - A		
	R[1]	Base Requirements
[1]Contact BTL for interim tests for this BIBB.		
Data Sharing - Lighting Output Status - A		
	R[1]	Base Requirements
[1]Contact BTL for interim tests for this BIBB.		
Data Sharing - Advanced Lighting Output - A		

	R[1]	Base Requirements
[1]Contact BTL for interim tests for this BIBB.		
Data Sharing - Lighting Output - B		
	R[1]	Base Requirements
[1]Contact BTL for interim tests for this BIBB.		
Data Sharing - Binary Lighting Output - B		
	R[1]	Base Requirements
[1]Contact BTL for interim tests for this BIBB.		
Data Sharing - Lighting Output Management - A		
	R[1]	Base Requirements
[1]Contact BTL for interim tests for this BIBB.		
Data Sharing - Lighting View - A		
	R[1]	Base Requirements
[1]Contact BTL for interim tests for this BIBB.		
Data Sharing - Lighting Advanced View - A		
	R[1]	Base Requirements
[1]Contact BTL for interim tests for this BIBB.		
Data Sharing - Lighting Modify - A		
	R[1]	Base Requirements
[1]Contact BTL for interim tests for this BIBB.		
Data Sharing - Lighting Advanced Modify - A		
	R[1]	Base Requirements
[1]Contact BTL for interim tests for this BIBB.		

[In BTL Checklist, modify the following BIBBs by adding the new sections, these new conditional sections should be added after the last Conditional but before the Optional section]

Device Management - Object Creation and Deletion - A		
…		
	C[2,6,7]	Can create and delete Staging objects
	C[2,6,7]	Can create and delete Audit Reporter objects
	C[2,6,7]	Can create and delete Audit Log objects
…		
... [7]Protocol_Revision 20 or higher must be claimed. [6]Contact BTL for interim tests for this object.		
Device Management - Object Creation and Deletion - B		
…		
	C[1,5,6]	Supports object creation and deletion of the Staging object
	C[1,5,6]	Supports object creation and deletion of the Audit Reporter object
	C[1,5,6]	Supports object creation and deletion of the Audit Log object
…		
... [6]Protocol_Revision 20 or higher must be claimed. [5]Contact BTL for interim tests for this object.		

[In BTL Checklist, add new section for Audit Reporting BIBBs]

Audit Reporting BIBBs

Audit Reporting - Logging - A		
	R[1]	Base Requirements
[1] Contact BTL for interim tests for this BIBB.		
Audit Reporting - Reporter - B		
	R[1]	Base Requirements
[1] Contact BTL for interim tests for this BIBB.		
Audit Reporting - Reporter - Simple - B		
	R[1]	Base Requirements
[1] Contact BTL for interim tests for this BIBB.		
Audit Reporting - Forwarder - B		
	R[1]	Base Requirements
[1] Contact BTL for interim tests for this BIBB.		
Audit Reporting - View - A		
	R[1]	Base Requirements
[1] Contact BTL for interim tests for this BIBB.		
Audit Reporting - Advanced View and Modify - A		
	R[1]	Base Requirements
[1] Contact BTL for interim tests for this BIBB.		

Section following was removed and placed in BTLWG-503 for further consideration.

BTL-15.1g-6: Correct Status_Flags Requirement in COMMAND_FAILURE Tests [BTLWG-434]

Overview:
BTL-CR-436 requested a change to step 2 of test 8.4.4.

Changes:

[In BTL Test Plan, Modify references to 135.1-2013 - 8.4.4 - COMMAND_FAILURE Tests]

5.2.11 Implements the COMMAND_FAILURE Algorithm

The IUT contains, or can be made to contain, an object that can generate ConfirmedEventNotifications and UnconfirmedEventNotifications with an Event_Type of COMMAND_FAILURE.

~~135.1-2013~~*BTL* - 8.4.4 - COMMAND_FAILURE Tests (ConfirmedEventNotification)	
Test Method	Manual
Configuration	As per ***~~ASHRAE 135.1-2013~~BTL Specified Tests***.
Test Conditionality	Must be executed.
Test Directives	This test must be repeated once for each object type that is capable of generating event notifications with an Event_Type of COMMAND_FAILURE.
Testing Hints	
Notes & Results	

5.3.9 Implements the COMMAND_FAILURE Algorithm

The IUT contains, or can be made to contain, an Event Enrollment object that can generate ConfirmedEventNotifications and UnconfirmedEventNotifications with an Event_Type of COMMAND_FAILURE.

~~135.1-2013~~*BTL* - 8.4.4 - COMMAND_FAILURE Tests (ConfirmedEventNotification)	
Test Method	Manual
Configuration	As per ***~~ASHRAE 135.1-2013~~BTL Specified Tests***.
Test Conditionality	Must be executed.
Test Directives	This test shall be executed with an Event Enrollment object that is configured to monitor a property in a device other than the IUT.
Testing Hints	
Notes & Results	

[In BTL Specified Tests, add the following tests to correct the versions in 135.1-2013.]

8.4.4 COMMAND_FAILURE Tests

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

BACnet Reference Clauses: 12.7, 12.12, 12.19, 13.2, 13.3.4, and 13.8.

Purpose: To verify the correct operation of the COMMAND_FAILURE algorithm.

Test Concept: The Feedback_Value (Feedback_Property_Reference) shall be decoupled from the input signal that is normally used to verify the output. Initially Present_Value (referenced property) and Feedback_Value (Feedback_Property_Reference) are in agreement. Present_Value (the referenced property) is changed and an event notification should be transmitted indicating a transition to an OFFNORMAL state. The Feedback_Value (Feedback_Property_Reference) is changed to again agree with the Present_Value (referenced property). A second event notification is transmitted indicating a return to a NORMAL state.

Configuration Requirements: The IUT shall be configured such that the Event_Enable property has a value of TRUE for the TO-OFFNORMAL and TO-NORMAL transitions. The Issue_Confirmed_Notifications property shall have a value of TRUE. The event-generating object shall be in a NORMAL state at the start of the test. The Feedback_Value property shall be decoupled from the input signal that is normally used to verify the output so that it can be independently manipulated.

In the test description below Present_Value is used as the referenced property and Feedback_Value is used to verify the output. If an Event Enrollment object is being tested these properties shall be replaced by the appropriate property reference.

Test Steps:

1. VERIFY Event_State = NORMAL
2. IF (the object being tested is not an Event Enrollment object) THEN
   VERIFY Status_Flags = ~~(FALSE, FALSE, FALSE, FALSE)~~*(FALSE, FALSE, ?, ?)*
3. IF (Present_Value is writable) THEN
   WRITE Present_Value = (a different value)
   ELSE
   MAKE (Present_Value take on a different value)
4. WAIT (Time_Delay)
5. BEFORE **Notification Fail Time**
   RECEIVE ConfirmedEventNotification-Request,

'Process Identifier' =	(any valid process ID),
'Initiating Device Identifier' =	IUT,
'Event Object Identifier' =	(the object being tested),
'Time Stamp' =	(the current local time),
'Notification Class' =	(the configured notification class),
'Priority' =	(the value configured to correspond to a TO-OFFNORMAL transition),
'Event Type' =	COMMAND_FAILURE,
'Notify Type' =	EVENT | ALARM,
'AckRequired' =	TRUE | FALSE,
'From State' =	NORMAL,
'To State' =	OFFNORMAL,
'Event Values' =	Present_Value, Status_Flags, Feedback_Value

6. TRANSMIT BACnet-SimpleACK-PDU
7. IF (the object being tested is not an Event Enrollment object) THEN
   VERIFY Status_Flags = (TRUE, FALSE, ?, ?)
8. VERIFY Event_State = OFFNORMAL
9. IF (Protocol_Revision is present and Protocol_Revision ≥ 1) THEN
   VERIFY Event_Time_Stamps = (the timestamp in step 5, *, *)
10. IF (Feedback_Value is writable) THEN
    WRITE Feedback_Value = (a value consistent with Present_Value)
    ELSE
    MAKE (Feedback_Value take on a value consistent with Present_Value)
11. WAIT (Time_Delay)
12. BEFORE **Notification Fail Time**
    RECEIVE ConfirmedEventNotification-Request,

'Process Identifier' =	(any valid process ID),
'Initiating Device Identifier' =	IUT,
'Event Object Identifier' =	(the object being tested),
'Time Stamp' =	(the current local time),
'Notification Class' =	(the configured notification class),
'Priority' =	(the value configured to correspond to a TO-NORMAL transition),
'Event Type' =	COMMAND_FAILURE,
'Notify Type' =	EVENT | ALARM,
'AckRequired' =	TRUE | FALSE,
'From State' =	OFFNORMAL,
'To State' =	NORMAL,
'Event Values' =	Present_Value, Status_Flags, Feedback_Value

13. TRANSMIT BACnet-SimpleACK-PDU
14. IF (the object being tested is not an Event Enrollment object) THEN
    VERIFY Status_Flags = (FALSE, FALSE, ?, ?)

15. VERIFY Event_State = NORMAL
16. IF (Protocol_Revision is present and Protocol_Revision ≥ 1) THEN
        VERIFY Event_Time_Stamps = (the timestamp in step 5, *, the timestamp in step 12)

Notes to Tester: The 'Message Text' parameter is omitted in the test description because it is optional. The IUT may include this parameter in the notification messages. The time stamps indicated by "*" in steps 9 and 16 can have a value that indicates an unspecified time or a time that precedes the timestamp in step 5.

BTL-15.1g-7: Optional Reliability in Global Group Tests [BTLWG-436]

Overview:
This document adds in the conditionality of the optional Reliability property of the Global Group object type.

Changes:

[In BTL Test Plan, change Global Group object test 7.3.2.13.X4]

BTL - 7.3.2.13.X4 - Present_Value Tracking and Reliability Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If the Reliability property is not present or cannot be made to not equal NO_FAULT_DETECTED, this test shall be skipped.
	Test Directives	The test shall be executed using a Global Group object.
	Testing Hints	
	Notes & Results	

[In BTL Specified Tests, change Global Group object tests 7.3.2.13.X2 and 7.3.2.13.X3]

7.3.2.13.X2 Reliability MEMBER_FAULT Test

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

Purpose: This test case verifies the FAULT flag of the Member_Status_Flags is TRUE and the Reliability property is equal to MEMBER_FAULT when a member of the Group_Members property goes into FAULT.

Test Concept: Force a member of the Group_Members property to enter a Fault condition and verify the Member_Status_Flags FAULT flag equals TRUE and Reliability equals MEMBER_FAULT.

Configuration Requirements: The IUT shall be configured with a Global Group object with the Group_Members property containing a member M1 at index N1 that has a value that can be made to indicate a fault condition (see Notes To Tester). The Out_Of_Service property of the Global Group object must remain FALSE throughout the test. W1 is the maximum time it takes for the Global Group to receive an update from M1.

Test Steps:

1. MAKE (M1 Status_Flags = {?, TRUE, ?, ?})
2. WAIT (W1)
3. VERIFY Member_Status_Flags = {?, TRUE, ?, ?}
4. IF (Reliability is present) THEN
   VERIFY Reliability = MEMBER_FAULT

Notes to Tester: Member_Status_Flags FAULT flag will the TRUE and the Reliability property will change to MEMBER_FAULT when a member of the Group_Members property goes into fault.

7.3.2.13.X3 Reliability COMMUNICATION_FAILURE Test

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

Purpose: This test case verifies that the Member_Status_Flags FAULT flag will remain FALSE while the Reliability property is COMMUNICATION_FAILURE.

Test Concept: Force a member of the Group_Members property to stop communicating and verify the Reliability property equals COMMUNICATION_FAILURE and the Member_Status_Flags FAULT flag remains FALSE.

Configuration Requirements: The IUT shall be configured with a Global Group object with the Group_Members containing a member M1 at index N1that can be made to discontinue communications and also respond with an error such as OBJECT/UNKNOWN_OBJECT. (See Notes To Tester). The Out_Of_Service property of the Global Group object must remain FALSE throughout the test. W1 is the maximum time it takes for the Global Group to receive an update from M1.

Test Steps:

1. MAKE (M1 fail (communications or error))
2. WAIT (W1)
3. IF (Reliability is present) THEN
   VERIFY Reliability = COMMUNICATION_FAILURE
4. VERIFY Member_Status_Flags = {?, FALSE, ?, ?}

Notes to Tester: Reliability will change to COMMUNICATION_FAILURE when a member is no longer able to communicate its Status_Flags property. This can occur when the device goes offline or the object is deleted within the device.

BTL-15.1g-8: Update Checklist to include FAULT_OUT_OF_RANGE algorithm [BTLWG-437]

Overview:

The checklist needs to be updated to contain the CHANGE_OF_RELIABILITY - FAULT_OUT_OF_RANGE algorithm option. This was introduced in Protocol_Revision 16.

Changes:

[In BTL Checklist, add the following entries to existing sections of the Alarm and Event Management Section]

Alarm and Event Management - Notification - A		
…		
	R[4,6]	Processes CHANGE_OF_RELIABILITY- FAULT_OUT_OF_RANGE notifications
[4]Required if the device implements protocol revision 16 or higher. … [6]Contact BTL for interim tests for this algorithm.		

Alarm and Event Management - Notification - Internal - B		
…		
	C[3,5,6]	Implements the CHANGE_OF_RELIABILITY - FAULT_OUT_OF_RANGE algorithm
…		
[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. … [5]Contact BTL for interim tests for this algorithm. [6]Protocol_Revision 16 or higher must be claimed.		
Alarm and Event Management - Notification - External - B		
…		
	C[1,2,3]	Implements the CHANGE_OF_RELIABILITY - FAULT_OUT_OF_RANGE algorithm
…		
[1] 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. [2]Contact BTL for interim tests for this algorithm. [3]Protocol_Revision 16 or higher must be claimed.		

BTL-15.1g-9: Fixed the COVU_Period and COVU_Recipient Zero Test [BTLWG-308]

Overview:

The wording in the test was incorrect in that some steps directed the tester to generate an UnconfirmedCOVNotification when the purpose of the test is to make sure one is not generated.

Changes:

[In BTL Specified Test, modify the existing test as follows]

7.3.2.13.X6 COVU_Period and COVU_Recipient Zero Test

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

Purpose: To verify that object O1 does not initiate UnconfirmedCOVNotification service requests when COVU_Period is zero or COVU_Recipient contains an empty list.

Test Concept: Configure O1 to produce unsubscribed UnconfirmedCOVNotifications, set COVU_Period to zero and ~~and~~ attempt to produce unsubscribed UnconfirmedCOVNotifications. Repeat with COVU_Recipients containing an empty list.

Configuration Requirements: At the start of the test, O1 shall be configured with a non-zero COVU_Period and a non-empty COV_Recipient property.

Test Steps:

1. MAKE (O1 issue an unsubscribed UnconfirmedCOVNotification)
2. BEFORE ~~Notification Fail Time~~ **Notification Fail Time**
   RECEIVE UnconfirmedCOVNotification-Request,
   DESTINATION = (any valid address),
   'Subscriber Process Identifier' = 0,
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = O1,
   'Time Remaining' = 0,
   'List of Values' = (any valid set of values)
3. ~~MAKE~~ *WRITE* (COVU_Period = 0)
4. MAKE (O1 *a condition that would normally cause the IUT to* issue an unsubscribed UnconfirmedCOVNotification)
5. WAIT ~~Notification Fail Time~~ **Notification Fail Time** times 2
6. CHECK (~~Verify~~ that O1 has not transmitted an UnconfirmedCOVNotification-Request~~.~~)
7. ~~MAKE~~ *WRITE* (COVU_Period <> 0)
8. MAKE (O1 issue an unsubscribed UnconfirmedCOVNotification)
9. BEFORE ~~Notification Fail Time~~ **Notification Fail Time**
   RECEIVE UnconfirmedCOVNotification-Request,
   DESTINATION = (any valid address),
   'Subscriber Process Identifier' = 0,
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = O1,
   'Time Remaining' = 0,
   'List of Values' = (any valid set of values)
10. ~~MAKE~~ *WRITE* (COVU_Recipient an empty list)
11. MAKE (O1 *a condition that would normally cause the IUT to* issue an unsubscribed UnconfirmedCOVNotification)
12. WAIT ~~Notification Fail Time~~ **Notification Fail Time** times 2
13. CHECK (~~Verify~~ that O1 has not transmitted an UnconfirmedCOVNotification-Request~~.~~)

BTL-15.1g-10: Fixed the Binary Object Elapsed_Active_Time Tests [BTLWG-441]

Overview:

BTLWG-39 was created and accepted by the BTL-WG to change the BTL specified test **7.3.1.9 Binary Object Elapsed Active Time Tests**. The change was to update/add the Test Concept, Configuration Requirement and Note To Tester to allow the Feedback_Value to be used for Elapsed_Active_Time tracking which was introduced in **Addendum 135-2010ad-3**. Initially the group thought we don't need to change anything in the Test Steps.

Changes:

[In BTL Specified Tests, revise existing test 7.3.1.9 as indicated.]

7.3.1.9 Binary Object Elapsed Active Time Tests

Reason For Change: During the review for the **addendum f to BTL Test Package 15.1**, it was found that the test steps should also updated for the IUT using Feedback_Value for the Elapsed_Active_Time calculation, and the test steps need to consider the time it takes for the Feedback_Value to update after the Present_Value changes. If the IUT uses Feedback_Value for the calculation then the test steps should check the Elapsed_Active_Time after the Feedback_Value changes, not after the Present_Value changes. Note To Tester was also moved from the bottom to right before the Test Steps

Purpose: To verify that the properties of binary objects that collectively track active time function properly. ~~If the Elapsed_Active_Time and Time_Of_Active_Time_Reset properties are not supported then this test shall be omitted.~~ This test applies to Binary Input, Binary Output, Binary Value *and Binary Lighting Output* objects.

Test Concept: The Present_Value *or Feedback_Value* of the binary object being tested is set to INACTIVE. The Elapsed_Active_Time property is checked to verify that it does not accumulate time while the object is in an INACTIVE state. The Present_Value *or Feedback_Value* is then set to ACTIVE. The Elapsed_Active_Time property is checked to verify that it is accumulating time while the object is in an ACTIVE state. The ~~Present_Value or Feedback_Value is then set to INACTIVE and the~~ Elapsed_Active_Time is reset. The Time_Of_Active_Time_Reset property is checked to verify that it has been updated.

Configuration Requirements: The object being tested shall be configured such that the Present_Value*, or Feedback_Value if that is used for the calculation,* and Elapsed_Active_Time properties are writable or another means of changing these properties shall be provided. *Whether Present_Value or Feedback_Value is used as the indicator for the calculation of the Elapsed_Active_Time is a local matter*.

Notes to Tester: This test intentionally allows devices to use Feedback_Value tracking specified in 135-2010ad-3 regardless of the Protocol_Revision claimed by the implementation.

Test Steps:

1. IF (Present_Value is writable) THEN
   WRITE Present_Value = INACTIVE
   VERIFY Present_Value = INACTIVE
   ELSE
   MAKE (Present_Value = INACTIVE)
2. *IF (Feedback_Value is used for Elapsed_Active_Time tracking)*
   WAIT (long enough for Feedback_Value, if present, to reflect the Present_Value)
   VERIFY Feedback_Value = INACTIVE
   ~~TRANSMIT ReadProperty-Request,~~
   ~~'Object Identifier' = (the object being tested),~~
   ~~'Property Identifier' = Elapsed_Active_Time~~
3. ~~RECEIVE ReadProperty-ACK,~~
   ~~'Object Identifier' = (the object being tested),~~
   ~~'Property Identifier' = Elapsed_Active_Time,~~

~~'Property Value' = (the elapsed active time, TELAPSED in seconds)~~

3. READ Elapsed_Active_Time = initialElapsedTime

-- verify that Elapsed_Active_Time does not change when the object is INACTIVE

4. WAIT (1 minute)
5. *VERIFY Elapsed_Active_Time = initialElapsedTime*

-- verify that Elapsed_Active_Time correctly reflects the time the object is ACTIVE

~~6. TRANSMIT ReadProperty-Request,~~
~~'Object Identifier' = (the object being tested),~~
~~'Property Identifier' = Elapsed_Active_Time~~
~~7. RECEIVE ReadProperty-ACK,~~
~~'Object Identifier' = (the object being tested),~~
~~'Property Identifier' = Elapsed_Active_Time,~~
~~'Property Value' = (the elapsed active time, $T_{ELAPSED}$ in seconds)~~

6. IF (Present_Value is writable) THEN
   WRITE Present_Value = ACTIVE
   VERIFY Present_Value = ACTIVE
   ELSE
   MAKE (Present_Value = ACTIVE)
7. *IF (Feedback_Value is used for Elapsed_Active_Time tracking)*
   WAIT (long enough for Feedback_Value, if present, to reflect the Present_Value)
   VERIFY Feedback_Value = ACTIVE

8. READ initialTime = (the IUT's Device object).Local_Time

9. WAIT (Internal Processing Fail Time + 30 seconds)

~~9. TRANSMIT ReadProperty-Request,~~
~~'Object Identifier' = (the object being tested),~~
~~'Property Identifier' = Elapsed_Active_Time~~
~~10. RECEIVE ReadProperty-ACK,~~
~~'Object Identifier' = (the object being tested),~~
~~'Property Identifier' = Elapsed_Active_Time,~~
~~'Property Value' = (T: ($T_{ELAPSED}$+30) <=T<=($T_{ELAPSED}$+30+Internal Processing Fail Time))~~

10. IF (Present_Value is writable) THEN
    WRITE Present_Value = INACTIVE
    VERIFY Present_Value = INACTIVE
    ELSE
    MAKE (Present_Value = INACTIVE)
11. *IF (Feedback_Value is used for Elapsed_Active_Time tracking)*
    WAIT (long enough for Feedback_Value, if present, to reflect the Present_Value)
    VERIFY Feedback_Value = INACTIVE
12. *READ currentTime = (the IUT's Device object).Local_Time*
13. *READ totalElapsedTime = Elapsed_Active_Time).Local_Time*
14. *CHECK (totalElapsedTime ~= (currentTime – initialTime) - initialElapsedTime)*

-- verify ability to reset Elapsed_Active_Time, if it is writable

15. *IF (Elapsed_Active_Time is writable) THEN*
    WRITE Elapsed_Active_Time = 0
    READ currentDate = (the IUT's Device object).Local_Date
    READ currentTime = (the IUT's Device object).Local_Time
    VERIFY Time_Of_Active_Time_Reset ~= {currentDate, currentTime}

~~12. IF (Elapsed_Active_Time is writable) THEN~~
~~WRITE Elapsed_Active_Time = 0~~
~~VERIFY Elapsed_Active_Time = 0~~
~~ELSE~~
~~MAKE (Elapsed_Active_Time = 0)~~
~~13. TRANSMIT ReadProperty-Request,~~
~~'Object Identifier' = (the IUT's Device object),~~
~~'Property Identifier' = Local_Date~~
~~14. RECEIVE ReadProperty-ACK,~~

~~'Object Identifier' = (the IUT's Device object),~~
~~'Property Identifier' = Local Date,~~
~~'Property Value' = (the current local date, D)~~
~~15. TRANSMIT ReadProperty Request,~~
~~'Object Identifier' = (the IUT's Device object),~~
~~'Property Identifier' = Local_Time~~
~~16. RECEIVE ReadProperty ACK,~~
~~'Object Identifier' = (the IUT's Device object),~~
~~'Property Identifier' = Local_Time,~~
~~'Property Value' = (the current local time, TLOC)~~
~~17. TRANSMIT ReadProperty Request,~~
~~'Object Identifier' = (the object being tested),~~
~~'Property Identifier' = Time_Of_Active_Time_Reset~~
~~18. RECEIVE ReadProperty ACK,~~
~~'Object Identifier' = (the object being tested),~~
~~'Property Identifier' = Present_ValueTime_Of_Active_Time_Reset,~~
~~'Property Value' = (a date and time such that the date = D and the time is approximately TLOC)~~

BTL-15.1g-11: Update Error Code references to include Error Class [BTLWG-145]

Overview:
Where Error Code is mentioned in 9.1.2.4 and 9.1.2.7 it should follow style convention by preceding that with mention that 'Error Class': SERVICES.

Changes:

[In BTL Specified Tests change the following sections]

9.1.2.4 Unsuccessful Alarm Acknowledgment of Confirmed Event Notifications Because the 'Event State Acknowledged' is Invalid
Reason for Change: This test was updated to account for revision 5 specifications. There is no new SSPC proposal.

Purpose: To verify that an alarm remains unacknowledged if the 'Event State Acknowledged' is inconsistent with the ~~other parameters~~ *Event_State* that ~~define~~ *was provided in the notification which is*~~the alarm~~ being acknowledged.

Test Concept: An alarm is triggered that causes the IUT to notify the TD and ~~at least~~ one other device. The TD acknowledges the alarm using an invalid event state and verifies that the acknowledgment is not accepted by the IUT and that the IUT does not notify other devices that the alarm was acknowledged. The TD then acknowledges the alarm using the proper event state and verifies that the acknowledgment is properly noted by the IUT. The IUT notifies all other recipients that the alarm was acknowledged.

Configuration Requirements: The IUT shall be configured with at least one object that can detect alarm conditions and send confirmed notifications. The Acked_Transitions property shall have the value B'111' indicating that all transitions have been acknowledged. The TD and ~~at least~~ one other BACnet device *if the IUT supports multiple recipients* shall be recipients of the alarm notification.

Test Steps: The test steps defined in 9.1.2.1 shall be followed except that in the first AcknowledgeAlarm request the 'Time Stamp' shall have the same value as the 'Time Stamp' from the event notification*, the 'To State' in the notification shall be any offnormal transition* and the 'Event State Acknowledged' shall have a*n offnormal* value that is different from the 'To State' in the event notification *and shall not be OFFNORMAL (2).*

Notes to Tester: A passing result is the same message sequence described *as the passing result* in 9.1.2.1 except that the *error reported shall have an Error Class of SERVICES and* Error Code ~~in step 7~~ ~~shall be~~ *of INVALID_EVENT_STATE. For devices claiming a Protocol Revision less than 5, Error Class of SERVICES and an Error Code of* INCONSISTENT_PARAMETERS *shall also be allowed. If the IUT can only be configured with one recipient in the Recipient_List property of the issuing Notification_class object, omit steps 5, 6, 15, and 16.*

9.1.2.7 Unsuccessful Alarm Acknowledgment of Unconfirmed Event Notifications Because the 'Event State Acknowledged' is Invalid
Reason for Change: This test was updated to account for revision 5 specifications. There is no new SSPC proposal. Made changes to allow cases where only one Recipient_List entry is supported.

Purpose: To verify that an alarm remains unacknowledged if the 'Event State Acknowledged' is inconsistent with the ~~other parameters~~*Event_State* that ~~define~~ *was provided in the notification which is* ~~the alarm~~ being acknowledged.

Test Concept: An alarm is triggered that causes the IUT to notify the TD and ~~at least~~ one other device. The TD acknowledges the alarm using an invalid 'Event State Acknowledged' and verifies that the acknowledgment is not accepted by the IUT and that the IUT does not notify other devices that the alarm was acknowledged. The TD then acknowledges the alarm using the proper 'Event State Acknowledged' and verifies that the acknowledgment is properly noted by the IUT. The IUT notifies all other recipients that the alarm was acknowledged.

Configuration Requirements: The IUT shall be configured with at least one object that can detect alarm conditions and send unconfirmed notifications. The Acked_Transitions property shall have the value B'111' indicating that all transitions have been acknowledged. The TD and ~~at least~~ one other BACnet device *if the IUT supports multiple recipients* shall be recipients of the alarm notification.

Test Steps: The test steps defined in 9.1.2.5 shall be followed except that in the first AcknowledgeAlarm request the 'Time Stamp' shall have the same value as the 'Time Stamp' from the event*, the 'To State' in the notification shall be any offnormal transition* and the 'Event State Acknowledged' shall have a*n offnormal* value that is different from the 'To State' in the event notification *and shall not be OFFNORMAL (2).*

Notes to Tester: A passing result is the same message sequence described as the passing result in 9.1.2.5 except that the *error reported shall have an Error Class of SERVICES and* Error Code ~~in step 7~~ ~~shall be~~ *of INVALID_EVENT_STATE. For devices claiming a Protocol Revision less than 5, an Error Class of SERVICES and Error Code of* INCONSISTENT_PARAMETERS *shall also be allowed. If the IUT can only be configured with one recipient in the Recipient_List property of the issuing Notification_class object, omit steps 4 and 12.*