BACnet® TESTING LABORATORIES ADDENDA

Addendum imp2 to BTL Test Package 23.3

Revision v3
Revised 3/29/2024

Approved by the BTL Working Group on January 18, 2024;
Approved by the BTL Working Group Voting Members on March 27, 2024;
Published on April 3, 2024.

[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 23.3 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-23.3 imp2-1: Improve Automation for Out_Of_Service Test [BTLWG-1395]

Overview:

This work item proposes changes that will make the test more automation friendly. The current version of the test modifies Present_Value during the test and expects that the Status_Flags will be unchanged at the end of the test. For objects that support intrinsic alarming or fault detection, it is possible that Status_Flags will change based on the final value of Present_Value. This work items adds steps to ensure Present_Value is restored to its original value before verifying Status_Flags at the end of the test.

Changes:

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, change references to test 7.3.1.1.1 from 135.1-2023 to BTL]

Specified Test Changes

[Move 7.3.1.1.1 from 135.1-2023 into BTL Specified tests, modify as shown.]

7.3.1.1.1 Out_Of_Service, Status_Flags, and Reliability Test

Reason for Change: Modified test to be more automation friendly.

Purpose: To verify that Present_Value is writable when Out_Of_Service is TRUE and that the interrelationship between the Out_Of_Service, Status_Flags, and Reliability properties.

Test Concept: The value of the Out_Of_Service property is set to TRUE and the Present_Value property is tested to be writable. The value of the Status_Flags property is validated and, if present, the value of the Reliability property is also validated. The value of the Status_Flags property, SF1, and, if present, the Reliability property, R1, are checked to ensure they return to their initial values when the value of the Out_Of_Service property is set to FALSE.

Configuration Requirements: If the selected object is commandable, the values of the entries in the Priority_Array above the selected priority, PTY1, shall be NULL.

Test Steps:

1. READ SF1 = Status_Flags
2. *READ PV1 = Present_Value*
3. IF Reliability is present THEN
   READ R1 = Reliability
4. IF (Out_Of_Service is writable) THEN
   WRITE Out_Of_Service = TRUE
   ELSE
   MAKE (Out_Of_Service TRUE)
5. VERIFY Out_Of_Service = TRUE
6. VERIFY Status_Flags = (?, ?, ?, TRUE)
7. REPEAT X = (all values meeting the functional range requirements of 7.2.1) DO {
   WRITE Present_Value, PTY1 = X
   VERIFY Present_Value = X
   }

8. WRITE Present_Value, PTY1 = PV1
~~*7*~~*9*. 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)
}
~~*8*~~*10*.IF (Out_Of_Service is writable) THEN
WRITE Out_Of_Service = FALSE
ELSE
MAKE (Out_Of_Service FALSE)
~~*9*~~*11*.VERIFY Out_Of_Service = FALSE
~~*10*~~*12*. VERIFY Status_Flags = SF1
~~*11*~~*13*. IF Reliability is present THEN
VERIFY Reliability = R1

BTL-23.3 imp2-2: Improve Test Concept for WPM Tests [BTLWG-1415]

Overview:

Test 9.23.2.X1 contains O1 is in the test but not defined in the Test Concept.

Test 9.23.2.X1 became 9.23.2.12 in 135.1-2023
Test 9.23.2.X5 became 9.23.2.16 in 135.1-2023
Test 9.23.2.X9 became 9.23.2.19 in 135.1-2023
Test 9.23.2.X10 became 9.23.2.20 in 135.1-2023
Test 9.23.2.X11 became 9.23.2.21 in 135.1-2023
Test 9.23.2.X12 became 9.23.2.22 in 135.1-2023

Changes:

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, change all references to 9.23.2.12 from 135.1-2023 to BTL]
[In BTL Test Plan, change all references to 9.23.2.16 from 135.1-2023 to BTL]
[In BTL Test Plan, change all references to 9.23.2.19 from 135.1-2023 to BTL]
[In BTL Test Plan, change all references to 9.23.2.20 from 135.1-2023 to BTL]
[In BTL Test Plan, change all references to 9.23.2.21 from 135.1-2023 to BTL]
[In BTL Test Plan, change all references to 9.23.2.22 from 135.1-2023 to BTL]

Specified Test Changes

[Move test 9.23.2.12 from 135.1-2023 into BTL Specified Tests and make the changes as noted.]

9.23.2.12 WritePropertyMultiple Reject Test
Reason for Change: Fix Test Concept to define O1 and O2.

Purpose: This test case verifies that the IUT does not send a Reject-PDU after applying part of a WritePropertyMultiple.

Test Concept: *Object, O1, containing writable property, P1 and object O2, containing writable property, P2*~~Two writable properties, P1 and P2~~ are written ~~to the IUT~~ but the portion of the WritePropertyMultiple specifying P2 is made invalid by omitting the 'Property Value' parameter. If the IUT returns a Reject, then the value of the first property is checked to ensure it has not changed.

Test Steps:

1. READ OldValue = O1, P1
2. TRANSMIT WritePropertyMultiple-Request,
   'Object Identifier' = O1,
   'Property Identifier' = P1,
   'Property Value' = (NewValue: any value other than OldValue that would be accepted by the IUT for P1)
   'Object Identifier' = O2,
   'Property Identifier' = P2
3. RECEIVE WritePropertyMultiple-Error,
   'Error Class' = SERVICES,
   'Error Code' = INVALID_TAG

'Object Identifier' = O2
'Property Identifier' = P2) |
RECEIVE BACnet-Reject-PDU,
'Reject Reason' = INVALID_TAG
| MISSING_REQUIRED_PARAMETER
| INCONSISTENT_PARAMETERS
| INVALID_PARAMETER_DATA_TYPE
| TOO_MANY_ARGUMENTS)
4. IF (a WritePropertyMultiple-Error was received in step 3) THEN
VERIFY (O1), P1 = NewValue
ELSE -- a Reject-PDU was received
VERIFY (O1), P1 = OldValue

[Move test 9.23.2.16 from 135.1-2023 into BTL Specified Tests and make the changes as noted.]

9.23.2.16 WritePropertyMultiple Reject Test for first element of 'List of Write Access Specifications'

Reason for Change: Add description of O1 in Test Concept.

Purpose: This test case verifies that if IUT does sends a Reject-PDU or Error-PDU then the write attempt for the remaining element of 'List of Write Access Specifications' do not take place.

Test Concept: *Object, O1, contains a writable property, P1 and object O2, contains a writable property, P2.* ~~Two writable properties,~~ P1 having value X and P2 having value Y are written to the IUT but the portion of the WritePropertyMultiple specifying P1 is made invalid by omitting the 'Property Value' parameter. The value of the properties are checked to ensure ~~that it has~~*they have* not changed.

Test Steps:

1. VERIFY (O1), P1= X
2. VERIFY (O2), P2=Y
3. TRANSMIT WritePropertyMultiple-Request,
'Object Identifier' = O1,
'Property Identifier' = P1,
'Object Identifier' = O2,
'Property Identifier' = P2
'Property Value' = (Any valid value not equal to Y))
4. RECEIVE WritePropertyMultiple-Error,
'Error Class' = SERVICES,
'Error Code' = INVALID_TAG
'Object Identifier' = O1
'Property Identifier' = P1) |
(RECEIVE BACnet-Reject-PDU,
'Reject Reason' = INVALID_TAG |
MISSING_REQUIRED_PARAMETER |
INCONSISTENT_PARAMETERS |
INVALID_PARAMETER_DATA_TYPE |
TOO_MANY_ARGUMENTS)
4. VERIFY (O1), P1 = X
5. VERIFY (O2), P2 = Y

[Move test 9.23.2.19 from 135.1-2023 into BTL Specified Tests and make the changes as noted.]

9.23.2.19 Date Non-Pattern Properties Test using WritePropertyMultiple Service

Reason for Change: Update Test Concept to include meaning of O1.

Purpose: To verify that the property being tested does not accept special date field values.

Test Concept: *O1 is the object being tested.* The property being tested, P1, is written with each of the special date field values to ensure that the property does not accept them. A date is selected which is within the date range that the IUT will accept for the property. The value, V1, written to the property is the date D1 with one of its fields replaced with one of the date special values. If the property is a complex datatype, the other fields in the value shall be set within the range accepted by the IUT. This test shall only be applied to devices claiming Protocol_Revision 11 or higher.

Notes to Tester: if P1 is an array, then a non-zero array index may be provided in the TRANSMIT and the same array index observed in the WritePropertyMultiple-Error.

Test Steps:
REPEAT SV = (year unspecified, month unspecified, day of month unspecified,
day of week unspecified, odd months, even months, last day of month,
even days, odd days) DO {
1. TRANSMIT WritePropertyMultiple-Request
'Object Identifier' = O1,
'Property Identifier' = P1,
'Property Value' = (V1 updated with the special value SV)
2. RECEIVE WritePropertyMultiple-Error,
'Error Class' = PROPERTY,
'Error Code' = VALUE_OUT_OF_RANGE,
'Object Identifier' = O1,
'Property Identifier' = P1)
| (BACnet-Reject-PDU
'Reject Reason' = INVALID_PARAMETER_DATATYPE)
| (BACnet-Reject-PDU
'Reject Reason'= INVALID_TAG)
}

[Move test 9.23.2.20 from 135.1-2023 into BTL Specified Tests and make the changes as noted.]

9.23.2.20 Time Non-Pattern Properties Test using WritePropertyMultiple Service

Reason for Change: Update Test Concept to include meaning of O1.

Purpose: To verify that the property being tested does not accept special time field values.

Test Concept: *O1 is the object being tested.* The property being tested, P1, is written with each of the special time field values to ensure that the property does not accept them. A time is selected which is within the time range that the IUT will accept for the property. The value, V1, written to the property is the time T1 with one of its fields replaced with one of the time special values. If the property is a complex datatype, the other fields in the value shall be set within the range accepted by the IUT. This test shall only be applied to devices claiming Protocol_Revision 11 or higher.

Notes to Tester: if P1 is an array, then a non-zero array index may be provided in the TRANSMIT and the same array index observed in the WritePropertyMultiple-Error.

Test Steps:
REPEAT SV = (hour unspecified, minute unspecified, second unspecified, hundredths unspecified) Do {
1. TRANSMIT WritePropertyMultiple-Request
'Object Identifier' = O1,
'Property Identifier' = P1,
'Property Value' = (V1 updated with the special value SV)
2. RECEIVE WritePropertyMultiple-Error,
'Error Class' = PROPERTY,
'Error Code' = VALUE_OUT_OF_RANGE,
'Object Identifier' = Object1,
'Property Identifier' = P1
| (BACnet-Reject-PDU
'Reject Reason' = INVALID_PARAMETER_DATATYPE)
| (BACnet-Reject-PDU
'Reject Reason'= INVALID_TAG)

}

[Move test 9.23.2.21 from 135.1-2023 into BTL Specified Tests and make the changes as noted.]

9.23.2.21 DateTime Non-Pattern Properties Test using WritePropertyMultiple Service

Reason for Change: Update Test Concept to include meaning of O1.

Purpose: To verify that the property being tested does not accept special date field values.

Test Concept: *O1 is the object being tested.* The property being tested, P_1, is written with each of the special datetime field values to ensure that the property does not accept them. A datetime DT_1 is selected which is within the range that the IUT will accept for the property. The value, V_1, written to the property is the datetime DT_1 with one of its fields replaced with one of the date or time special values. If the property is a complex datatype, the other fields in the value shall be set within the range accepted by the IUT. This test shall only be applied to devices claiming Protocol_Revision 11 or higher.

Notes to Tester: if P1 is an array, then a non-zero array index may be provided in the TRANSMIT and the same array index observed in the WritePropertyMultiple-Error.

Test Steps:
REPEAT SV = (year unspecified, month unspecified, day of month unspecified, day of week unspecified,
odd months, even months, last day of month, even days, odd days, hour unspecified, minute unspecified, second unspecified, hundredths unspecified) DO {
1. TRANSMIT WritePropertyMultiple-Request,
'Object Identifier' = O1,
'Property Identifier' = P1,
'Property Value' = (DT_1 updated with the special value SV)
2. RECEIVE WritePropertyMultiple-Error,
'Error Class' = PROPERTY,
'Error Code' = VALUE_OUT_OF_RANGE,
'Object Identifier' = Object1,
'Property Identifier' = P1)
| (BACnet-Reject-PDU
'Reject Reason' = INVALID_PARAMETER_DATATYPE)
| (BACnet-Reject-PDU
'Reject Reason'= INVALID_TAG)
}

[Move test 9.23.2.22 from 135.1-2023 into BTL Specified Tests and make the changes as noted.]

9.23.2.22 BACnetDateRange Non-Pattern Properties Test using WritePropertyMultiple Service

Reason for Change: Update Test Concept to include meaning of O1.

Purpose: To verify that the property being tested does not accept special date field values, except for fully unspecified start of the range or fully unspecified end of the range, or both.

Test Concept: *O1 is the object being tested.* A BACnetDateRange property, or property that is a complex datatype containing BACnetDateRange P1 is written with each of the special field values to ensure that the property does not accept them. Each half of the dateRange DR1 is selected so it is within the range that the IUT will accept for the property. The value, V1 written to the property is the dateRange DR1 with one of its fields replaced with one of the date special values. If the property is a complex datatype, the other fields in the value shall be set within the range accepted by the IUT. This test shall only be applied to devices claiming Protocol_Revision 11 or higher.

Notes to Tester: if P1 is an array, then a non-zero array index may be provided in the TRANSMIT and the same array index observed in the WritePropertyMultiple-Error.

Test Steps:
REPEAT SV = (year unspecified, month unspecified, day of month unspecified,
day of week unspecified, odd months, even months, last day of month,

even days, odd days) DO {
1. TRANSMIT WritePropertyMultiple-Request,
'Object Identifier' = O1,
'Property Identifier' = P1,
'Property Value' = (DR1 with startDate updated with special value SV)
2. RECEIVE WritePropertyMultiple-Error,
'Error Class' = PROPERTY,
'Error Code' = VALUE_OUT_OF_RANGE,
'Object Identifier' = Object1,
'Property Identifier' = P1
| (BACnet-Reject-PDU
'Reject Reason' = INVALID_PARAMETER_DATATYPE)
| (BACnet-Reject-PDU
'Reject Reason'= INVALID_TAG)
3. TRANSMIT WritePropertyMultiple-Request,
'Object Identifier' = O1,
'Property Identifier' = P1,
'Property Value' = (DR1 with endDate updated with special value SV)
4. RECEIVE WritePropertyMultiple-Error,
'Error Class' = PROPERTY,
'Error Code' = VALUE_OUT_OF_RANGE,
'Object Identifier' = Object1,
'Property Identifier' = P1)
| (BACnet-Reject-PDU
'Reject Reason' = INVALID_PARAMETER_DATATYPE)
| (BACnet-Reject-PDU
'Reject Reason'= INVALID_TAG)
}

BTL-23.3 imp2-3: Virtual Routing and GW-VN-B [BTLWG-1421, CR-0473]

Overview:

Ensure that, for IUTs that support routing to a virtual network but do not claim the GW-VN-B, testing of virtual devices is performed.

Changes:

Checklist Changes

None

Test Plan Changes

[change Test Plan Clause 10.1.3]

10.1.3 Routes Packets Between a Physical LAN and One or More Virtual LANs

The device can route BACnet packets between a physical BACnet LAN and one or more virtual BACnet LANs that contain one or more virtual BACnet devices. See H.1 and H.2 in the BACnet standard for a description of virtual BACnet LANs and virtual BACnet devices.

Verify Virtual Devices		
	Test Conditionality	Must be executed
	Test Directives	Test the virtual devices as per their BTL Checklist.
	Testing Hints	Similar to testing derivative products, the functionality supported might need to be spread over 2 or more virtual devices. In such cases, to which of the virtual devices any particular test is applied is left up to the tester as long as all applicable tests are executed.
…		

[change Test Plan Clause 10.2]

10.2 Network Management - Router Configuration - B

The tests are designed for testing routing devices which connect two or more BACnet networks ~~at the network layer~~ or routing devices which connect a BACnet network to one or more virtual BACnet networks.

…

[change Test Plan Clause 11.1.1]

11.1 Gateway - Virtual Network - B

11.1.1 Base Requirements

The IUT supports routing to virtual networks.

~~**Verify Virtual Devices**~~		
	~~**Test Conditionality**~~	~~Must be executed~~
	~~**Test Directives**~~	~~Test the virtual devices as per their BTL Checklist.~~
	~~**Testing Hints**~~	~~Similar to testing derivative products, the functionality supported might need to be spread over 2 or more virtual devices. In such cases,~~

		~~to which of the virtual devices any particular test is applied is left up to the test as long as all applicable tests are executed.~~
Verify Checklist		
	Test Conditionality	Must be executed
	Test Directives	Verify that the IUT claims the Network Management - Routing option "Routes Packets Between a Physical LAN and One or More Virtual LANs"
	Testing Hints	
Verify Checklist		
	Test Conditionality	Must be executed
	Test Directives	Verify the IUT claims support for NM-RC-B.
	Testing Hints	

Specified Test Changes

None

BTL-23.3 imp2-4: Reliability_Evaluation_Inhibit Test [BTLWG-1430]

Overview:

This proposal revisits how the Reliability_Evaluation_Inhibit property is tested due to the historical clarification requests that have been created. (CR-0452, CR-0455, CR-0523).

Issues to be resolved:

- A device is allowed to support Reliability_Evaluation_Inhibit and reliability-evaluation without support event generation. See Clause 12.1.8.
- Reliability_Evaluation_Inhibit property tests could not be run without claiming AE-N-I-B.
- Ensure Reliability_Evaluation_Inhibit is not claimed if the IUT can't enter fault without change the Reliability property.

The proposed solution is to test reliability-evaluation on a per object basis instead of part of AE-N-I-B.

Changes:

Checklist Changes

[In Clause 3, add to each object type listed: analog-input, analog-output, analog-value, binary-input, binary-output, binary-value, calendar, command, device, event-enrollment, loop, multi-state-input, multi-state-output, notification-class, program, schedule, multi-state-value, trend-log, life-safety-point, life-safety-zone, accumulator, pulse-converter, event-log, global-group, trend-log-multiple, load-control, access-door, timer, access-credential, access-point, access-rights, access-user, access-zone, credential-data-input, bitstring-value, characterstring-value, datepattern-value, date-value, datetimepattern-value, datetime-value, integer-value, large-analog-value, octetstring-value, positive-integer-value, timepattern-value, time-value, notification-forwarder, channel, lighting-output, binary-lighting-output, network-port, escalator, lift, staging, audit-log, audit-reporter]

Note: Object types of file, group, averaging, structured-view, alert-enrollment, and elevator-group do not receive these changes.

[object type] Object		
…		
	O[x]	Supports configurable Reliability_Evaluation_Inhibit property
[x] Protocol_Revision 13 or higher must be claimed. IUT shall not claim this functionality if the only method of generating a fault condition is by writing to the Reliability property.		

Alarm and Event Management - Notification - Internal - B		
…		
	O[x]	Implements the Reliability_Evaluation_Inhibit property with Fault Reporting
[x] Protocol_Revision 13 or higher must be claimed. IUT shall not claim this functionality if the only method of generating a fault condition is by writing to the Reliability property.		

Test Plan Changes

[In Clause 3, add to each object type listed: analog-input, analog-output, analog-value, binary-input, binary-output, binary-value, calendar, command, device, event-enrollment, loop, multi-state-input, multi-state-output, notification-class, program, schedule, multi-state-value, trend-log, life-safety-point, life-safety-zone, accumulator, pulse-converter, event-log, global-group, trend-log-multiple, load-control, access-door, timer, access-credential, access-point, access-rights, access-user, access-zone, credential-data-input, bitstring-value, characterstring-value, datepattern-value, date-value, datetimepattern-value, datetime-value, integer-value, large-analog-value, octetstring-value, positive-integer-value, timepattern-value, time-value, notification-forwarder, channel, lighting-output, binary-lighting-output, network-port, escalator, lift, staging, audit-log, audit-reporter]

Note: Object types of file, group, averaging, structured-view, alert-enrollment, and elevator-group do not receive these changes.

3.x.y Supports Configurable Reliability_Evaluation_Inhibit Property

The Reliability_Evaluation_Inhibit property for this object type, contained in the IUT are either writable or can be modified by any other means.

BTL - 7.3.1.21.X1 - Reliability_Evaluation_Inhibit Object Test		
	Test Conditionality	This test can be skipped if this object supports fault reporting.
	Test Directives	
	Testing Hints	

5.2.43 Implements the Reliability_Evaluation_Inhibit Property with Fault Reporting

The IUT supports the Reliability_Evaluation_Inhibit property to control the reliability evaluation in objects and supports fault reporting.

~~135.1-2023~~**BTL - 7.3.1.21.1 - Reliability_Evaluation_Inhibit Test**		
	Test Conditionality	Must be executed. ~~If Protocol_Revision < 13, then this test shall be skipped.~~ ~~If no object exists in the IUT for which fault conditions can be generated or has no object in which Reliability_Evaluation_Inhibit can be made TRUE then this test shall be skipped.~~
	Test Directives	This test must be repeated once for each object type that supports fault conditions and fault reporting ~~The objects selected by the tester should include all variants that differ in the set of supported alarming properties, or the writability of any of those properties. At least one instance of each variant should be selected.~~
	Testing Hints	
135.1-2023 - 7.3.1.21.2 - Reliability_Evaluation_Inhibit Summarization Test		
	Test Conditionality	Must be executed. ~~If Protocol_Revision < 13, then this test shall be skipped.~~ ~~If no object exists in the IUT for which fault conditions can be or has no object in which Reliability_Evaluation_Inhibit can be made TRUE then this test shall be skipped.~~
	Test Directives	This test must be repeated once for each object type that supports fault conditions and fault reporting ~~The objects selected by the tester should include all variants that differ in the set of supported alarming properties, or the writability of any of those properties. At least one instance of each variant should be selected.~~

Specified Test Changes

[Move test 7.3.1.21.1 from 135.1 to BTL Specified Tests and modify as shown below.]

7.3.1.21.1 Reliability_Evaluation_Inhibit Test

Reason for Change: Removed conditional event reporting and the conditionality that allows the test to be skipped.

Purpose: To verify that Reliability_Evaluation_Inhibit controls whether or not fault conditions are detected *and events are generated*.

Test Concept: Select an event generating object, O1, which supports the Reliability_Evaluation_Inhibit property. With Reliability_Evaluation_Inhibit FALSE, make a fault condition exist. Verify that Reliability changes and that a notification is generated. Set Reliability_Evaluation_Inhibit to TRUE. Verify that the Reliability changes to NO_FAULT_DETECTED and that a TO_NORMAL notification is generated. Remove the fault condition and ensure that no notification is generated. Make a fault condition exist and verify that Reliability remains NO_FAULT_DETECTED, and that no notification is generated.

Configuration Requirements: O1 is configured to detect and report unconfirmed events, is in the NORMAL state, and Reliability_Evaluation_Inhibit equals FALSE, so that reliability evaluation for that object is configured to detect fault conditions. ~~If no object exists in the IUT for which fault conditions can be generated then this test shall be skipped.~~

Notes to Tester: This behavior can alternately be tested using the ConfirmedEventNotification service, but it is not necessary to test both.

Test Steps:

1. VERIFY pCurrentState = NORMAL
2. VERIFY Reliability = NO_FAULT_DETECTED
3. MAKE(~~a fault condition exist for O1~~*a condition exist that would cause O1 to generate a TO_FAULT transition*)
4. ~~IF the IUT supports event reporting THEN~~
   BEFORE **Notification Fail Time**
   RECEIVE UnconfirmedEventNotification-Request
   'Process Identifier' = (the value configured for the transition),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = O1,
   'Time Stamp' = (any valid timestamp),
   'Priority' = (any valid priority),
   'Event Type' = CHANGE_OF_RELIABILITY,
   'Message Text' = (optional, any valid message text),
   'Notify Type' = ALARM | EVENT,
   'AckRequired' = TRUE | FALSE,
   'From State' = NORMAL,
   'To State' = FAULT,
   'Event Values' = (any values appropriate to CHANGE_OF_RELIABILITY)
5. VERIFY Reliability <> NO_FAULT_DETECTED
6. *VERIFY pCurrentState = FAULT*
7. IF Reliability_Evaluation_Inhibit is writable THEN
   WRITE Reliability_Evaluation_Inhibit = TRUE
   ELSE
   MAKE(Reliability_Evaluation_Inhibit TRUE)
8. ~~IF the IUT supports event reporting THEN~~
   BEFORE **Internal Processing Fail Time** + **Notification Fail Time**
   RECEIVE UnconfirmedEventNotification-Request
   'Process Identifier' = (the value configured for the transition),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = O1,
   'Time Stamp' = (any valid timestamp),
   'Priority' = (any valid priority),
   'Event Type' = CHANGE_OF_RELIABILITY,
   'Message Text' = (optional, any valid message text),
   'Notify Type' = ALARM | EVENT,
   'AckRequired' = TRUE | FALSE,
   'From State' = FAULT,
   'To State' = NORMAL,
   'Event Values' = (any values appropriate to CHANGE_OF_RELIABILITY)
9. VERIFY Reliability = NO_FAULT_DETECTED
10. VERIFY pCurrentState = NORMAL
11. MAKE(remove the fault condition)
12. WAIT(pTimeDelayNormal)
13. WAIT **Notification Fail Time**

14. CHECK (that the IUT did not send any event notifications for O1)
15. VERIFY Reliability = NO_FAULT_DETECTED
16. MAKE(~~a fault condition exist for O1~~*a condition exist that would cause O1 to generate a TO_NORMAL transition*)
17. WAIT **Notification Fail Time**
18. VERIFY Reliability = NO_FAULT_DETECTED
19. VERIFY pCurrentState = NORMAL
20. CHECK (that the IUT did not send any event notifications for O1)

[Add new test 7.3.1.21.X1]

7.3.1.21.X1 Reliability_Evaluation_Inhibit Object Test

Reason for Change: No test for this functionality.

Purpose: To verify that Reliability_Evaluation_Inhibit controls whether or not an object performs the reliability-evaluation process.

Test Concept: Select an object, O1, which supports the Reliability_Evaluation_Inhibit property. With Reliability_Evaluation_Inhibit FALSE, make a fault condition exist. Verify that the Reliability property changes. Set Reliability_Evaluation_Inhibit to TRUE. Verify that the Reliability changes to NO_FAULT_DETECTED. Make a fault condition exist and verify that Reliability property remains NO_FAULT_DETECTED.

Configuration Requirements: The Event_State of O1 is NORMAL, and Reliability_Evaluation_Inhibit equals FALSE.

Test Steps:

1. VERIFY Event_State = NORMAL
2. VERIFY Reliability = NO_FAULT_DETECTED
3. MAKE(a fault condition exist for O1)
4. VERIFY Reliability <> NO_FAULT_DETECTED
5. VERIFY Event_State = FAULT
6. IF Reliability_Evaluation_Inhibit is writable THEN
   WRITE Reliability_Evaluation_Inhibit = TRUE
   ELSE
   MAKE(Reliability_Evaluation_Inhibit TRUE)
7. VERIFY Reliability = NO_FAULT_DETECTED
8. VERIFY Event_State = NORMAL
9. MAKE(remove the fault condition)
10. VERIFY Reliability = NO_FAULT_DETECTED
11. MAKE(a fault condition exist for O1)
12. VERIFY Reliability = NO_FAULT_DETECTED
13. VERIFY Event_State = NORMAL

BTL-23.3 imp2-5: AE-N-E-B Should Reference Event Enrollment Object [BTLWG-1434]

Overview:

The opening paragraphs in several sections in AE-N-E-B do not specify the use of an Event Enrollment object.

Changes:

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, modify sections under 5.3 AE-N-E-B as noted below.]

5.3Alarm and Event Management - Notification - External - B

…

5.3.5 Implements the CHANGE_OF_BITSTRING Algorithm

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

…

5.3.6 Implements the CHANGE_OF_STATE Algorithm

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

…

5.3.7 Implements the Numeric Form of the CHANGE_OF_VALUE Algorithm

The IUT contains, or can be made to contain, an Event Enrollment object that can generate ~~ConfirmedEventNotifications and UnconfirmedEventNotifications~~EventNotifications with an Event_Type of CHANGE_OF_VALUE where the monitored value is of data type Real.

…

5.3.8 Implements the Bit String Form of the CHANGE_OF_VALUE Algorithm

The IUT contains, or can be made to contain, an Event Enrollment object that can generate ~~ConfirmedEventNotifications and UnconfirmedEventNotifications~~EventNotifications with an Event_Type of CHANGE_OF_VALUE where the monitored value is of datatype bit string.

…

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~~EventNotifications with an Event_Type of COMMAND_FAILURE.

…

5.3.10 Implements the FLOATING_LIMIT Algorithm

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

…

5.3.11 Implements the OUT_OF_RANGE Algorithm

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

…

5.3.12 Implements the DOUBLE_OUT_OF_RANGE Algorithm

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

…

5.3.13 Implements the SIGNED_OUT_OF_RANGE Algorithm

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

…

5.3.14 Implements the UNSIGNED_OUT_OF_RANGE Algorithm

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

…

5.3.15 Implements the CHANGE_OF_CHARACTERSTRING Algorithm

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

…

5.3.16 Implements the CHANGE_OF_STATUS_FLAGS Algorithm

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

…

…

5.3.27 Implements the ACCESS_EVENT Algorithm

The IUT contains, or can be made to contain, an Event Enrollment object that can generate *EventNotifications with an* Event_Type of ACCESS_EVENT.

…

Specified Test Changes

None

BTL-23.3 imp2-6: NPO Volatility Test [BTLWG-1435]

Overview:

The use of "MAKE (the IUT power cycle to reinitialize)" in step 4 infers the device is power cycled to accept the changes to the NPO. The Changes_Pending property should be used to check if changes need to be saved.

Changes:

Checklist Changes

None

Test Plan Changes

None

Specified Test Changes

[Move test 7.3.2.46.1.3 from 135.1-2023 to BTL Specified Tests and modify as shown here.]

7.3.2.46.1.3 Network Port Non-Volatility Properties Test

Reason for Change: Modify the test to save changes via normal means before verifying the Power Cycle keeps the changes.

Purpose: This test verifies that after Network Port properties are changed, and activated, the revised value is maintained through a power failure and device restart.

Test Concept: Write one or more properties, P1 ... PN, of a Network Port object which are required for proper operation of the network port. If any of the properties utilize the pending changes functionality, activate the changes. Restart the IUT device by temporarily removing power. When the device has resumed operation after that restart, verify that the new values for the properties were maintained across the reset and are in use by the port.

Test Steps:

1. REPEAT P = P1 ... PN {
   WRITE P = (a new value different from the property's current value)
   }
2. IF *(Changes_Pending is TRUE)* ~~any of the properties utilize the pending change functionality~~ THEN
   ~~VERIFY Changes_Pending = TRUE~~
   TRANSMIT ReinitializeDevice-Request
   'Reinitialized State of Device' = WARMSTART | ACTIVATE_CHANGES
   'Password' = (any valid password)
   RECEIVE BACnet-SimpleACK-PDU
   MAKE(reconfigure the TD and other devices on the network to the new network settings)
   WAIT Activate Changes Fail Time
   ~~ELSE~~
   VERIFY Changes_Pending = FALSE
3. REPEAT P = P1 ... PN {
   VERIFY P = (the new value for the property)
   }
4. MAKE (the IUT power cycle ~~to reinitialize~~)
5. REPEAT P = P1 ... PN {
   VERIFY P = (the new value for the property)
   CHECK (that the value for P is in use by the network port)
   }

BTL-23.3 imp2-7: Change Life Safety Checklist [BTLWG-1437]

Overview:

The Checklist for the CHANGE_OF_LIFE_SAFETY Algorithm in AE-N-I-B and AE-N-E-B contain footnotes stating "Contact BTL for interim tests for the algorithm."

Changes:

Checklist Changes

[Remove CHANGE_OF_LIFE_SAFETY and ACCESS_EVENT from AE-N-I-B]

Alarm and Event Management - Notification - Internal - B		
	R	Base Requirements
…		
	~~C[3,5]~~	~~Implements the CHANGE_OF_LIFE_SAFETY algorithm~~
	~~O~~	~~Implements the ACCESS_EVENT algorithm~~
…		
[1] Required if EventNotifications with service parameter AckRequired = True can be issued. [2] At least one of these options must be supported to claim support for this BIBB. [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. [4] At least one of these options must be supported to claim support for this BIBB. The BACnetDateTime form of the timestamp is the recommended option. ~~[5] Contact BTL for interim tests for this algorithm.~~ [6] Protocol_Revision 16 or higher must be claimed. [7] Protocol_Revision 17 or higher must be claimed. [8] Protocol_Revision 18 or higher must be claimed. [9] A device shall support CHANGE_OF_RELIABILITY in any object which generates event notifications and in which the Reliability property can take on a value other than NO_FAULT_DETECTED.		

[Remove CHANGE_OF_LIFE_SAFETY and ACCESS_EVENT from AE-N-E-B]

Alarm and Event Management - Notification - External - B		
	R	Base Requirements
…		
	~~C[1,2]~~	~~Implements the CHANGE_OF_LIFE_SAFETY algorithm~~
	~~O~~	~~Implements the ACCESS_EVENT algorithm~~
…		
[1] 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. ~~[2] Contact BTL for interim tests for this algorithm.~~ [3] Protocol_Revision 16 or higher must be claimed. [4] Protocol_Revision 17 or higher must be claimed. [5] Protocol_Revision 18 or higher must be claimed. [6] A device shall support CHANGE_OF_RELIABILITY in any object which generates event notifications and in which the Reliability property can take on a value other than NO_FAULT_DETECTED.		

Test Plan Changes

[Remove sections 5.2.38 and 5.2.39 from AE-N-I-B and renumber following sections]

[Remove sections 5.3.26 and 5.3.27 from AE-N-E-B and renumber following sections]

[Change sections in AE-LS-B to be specific about testing a Life Safety Object]

5.22.4 Implements Intrinsic Alarming

The IUT contains, or can be made to contain, a Life Safety~~n object other than an Event Enrollment~~ object that can generate ConfirmedEventNotifications and UnconfirmedEventNotifications.

Verify ~~Checklist~~Test Selection		
	Test Conditionality	Must be executed.
	Test Directives	Ensure this functionality is tested on ~~non Event Enrollment~~Life Safety objects by the clause 8.4 or 8.5 algorithm tests listed later in this section.
	Testing Hints	

5.22.5 Supports the Event Enrollment Object Using the CHANGE_OF_LIFE_SAFETY Algorithm

The IUT contains, or can be made to contain, an Event Enrollment object that can generate ConfirmedEventNotifications and UnconfirmedEventNotifications for the CHANGE_OF_LIFE_SAFETY algorithm.

Verify ~~Checklist~~Test Selection		
	Test Conditionality	Must be executed.
	Test Directives	Ensure this functionality is tested on Event Enrollment objects by the clause 8.4 or 8.5 algorithm tests listed later in this section. ~~Verify that the IUT claims support for CHANGE_OF_LIFE_SAFETY in the EventEntrollment Objects in AE-N-I-B.~~
	Testing Hints	

Specified Test Changes

None

BTL-23.3 imp2-8: Remove ReadOnly Properties from DM-LM-A [BTLWG-1453, CR-0556]

Overview:

In functionality checklist, under DM-LM-A we could see support for read-only properties check name

After receiving AddListElement and RemoveListElement request for above yellow highlighted Read-Only properties TD will always respond with “Error Class = PROPERTY, Error Code = WRITE_ACCESS_DENIED” instead of BACnet-SimpleACK-PDU.

A device can claim just the "Supports adding and removing entries in proprietary list properties of primitive datatypes" to claim this BIBB.

Solution:
Read-Only properties should not be present in the functionality checklist.
A device should not be able to claim only "Supports adding and removing entries in proprietary list properties of primitive datatypes" to claim this BIBB.

Changes:

Checklist Changes

[Remove sections from DM-LM-A as shown below]

Device Management - List Manipulation - A		
	R	Base Requirements
	~~C~~[1]	~~Supports adding and removing Device / Device_Address_Binding entries~~
	~~C~~[1]	~~Supports adding and removing Device / Active_COV_Subscriptions entries~~
	~~C~~[1]	~~Supports adding and removing Device / Slave_Address_Binding entries~~
	~~C~~[1] *O*	Supports adding and removing entries in proprietary list properties of primitive datatypes
	~~C~~[1,2]	~~Supports adding and removing Network Port / Slave_Address_Binding entries~~
	~~C~~[1,3]	~~Supports adding and removing Device / Active_COV_Multiple_Subscriptions entries~~
[1] At least one of these options must be supported to claim support for this BIBB. [2] Protocol_Revision 17 or higher must be claimed. [3] Protocol_Revision 18 or higher must be claimed.		

Test Plan Changes

[Remove sections: 8.23.8, 8.23.9, 8.23.11, 8.23.50, 8.23.53 from DM-LM-A (8.23)]

Specified Test Changes

None

BTL-23.3 imp2-9: ReadRange Negative Test Changes [BTLWG-1456]

Overview:

During execution of the test package, it was noted that the test conditionality for 9.21.2.X7 and 9.21.2.X8 was overly restrictive. The tests and test directives are being modified to allow these tests to be run against a larger set of properties.

Changes:

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, section 4.16.1 change the Test Conditionality and Testing Hints]

…		
BTL - 9.21.2.X7 - Attempting to Read a List Property by Sequence That Does not Have Sequence Numbers		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision 21 or greater~~and supports a List Property that supports Timestamps.~~
	Test Directives	
	Testing Hints	Can be executed against the Device_Address_Binding property of Device object.
BTL - 9.21.2.X8 - Attempting to Read a List Property by ByTime That Does not Have Timestamps		
	Test Conditionality	Must be executed ~~if the IUT claims Protocol_Revision 21 or greater and supports a List Property that supports Sequence Numbers~~.
	Test Directives	
	Testing Hints	Can be executed against the Device_Address_Binding property of Device object.

Specified Test Changes

[In BTL Specified Tests, change test 9.21.2.X8. Note the ELSE from step two is being added back into BTL Specified Tests version since it is missing in BTL Specified Tests. This ELSE was not added in this work item. It already existed.]

9.21.2.X8 Attempting to Read a List Property by ByTime That Does not Have Timestamps

Reason for Change: No test exists for this functionality. Tests added as per 135-2016-bu1

References: 15.8.1.3.1, 18.3

Purpose: To verify the correct execution of the ReadRange service request when the requested property does not support timestamps.

Test Concept: A ReadRange request is transmitted by the TD requesting a specified reference time and count of items. The IUT shall respond with an error~~by returning the appropriate error code. This test is only applied to devices with a Protocol_Revision of 21 or higher~~.

Test Configuration: A list property that does not support timestamps must be selected for this test. ~~If no suitable property exists in the device, this test shall be skipped.~~

Test Steps:

1. TRANSMIT Read-Range-Request,
   'Object Identifier' = (the object configured for this test),
   'Property Identifier' = (the list property configured for this test),
   'Reference Time' = (any valid specific BACnetDateTime)
   'Count' = (any valid value)
2. RECEIVE BACnet-Error-PDU,
   'Error Class' = PROPERTY,
   'Error Code' = LIST_ITEM_NOT_TIMESTAMPED

   ELSE

   RECEIVE BACnet-Error-PDU
   'Error Class' = E (any valid Error Class)
   'Error Code' = (any valid Error Code for class E)

BTL-23.3 imp2-10: Split TO_FAULT from Acked_Transitions Test [BTLWG-1471]

Overview:

Splitting the to-fault test in 7.3.1.11.1 allows for a much clearer Test Plan.

For 7.3.1.11.2 "Acked_Transitions Test for Latching Objects" requires the object to be preconfigured in a non-NORMAL state. It does not deal with pTimeDelayNormal for to-NORMAL transitions. This functionality can be tested using 7.3.1.11.1 since "MAKE (pMonitoredValue have a value that is NORMAL)" in Step 11 allows for the use of LifeSafetyOperation Service to reset pMonitoredValue and 7.3.1.11.X2 can test the to-fault condition.

Changes:

Checklist Changes

None

Test Plan Changes

…		
135.1-2023~~BTL~~ - 7.3.1.11.1 - Acked_Transitions Test		
	Test Conditionality	Must be executed if the IUT supports normal to offnormal event generating objects. ~~If the IUT only supports event generating objects which latch their event state and require an acknowledgement before unlatching, this test shall be skipped.~~ ~~Only life safety objects are allowed to latch in this manner.~~
	Test Directives	
	Testing Hints	
~~BTL - 7.3.1.11.2 - Acked_Transitions Test for Latching Objects~~		
	~~Test Conditionality~~	~~If the IUT does not support event generating objects which latch their event state and require an acknowledgement before unlatching, this test shall be skipped.~~ ~~Only life safety objects are allowed to latch in this manner.~~
	~~Test Directives~~	~~Apply the test for each supported transition.~~
	~~Testing Hints~~	
BTL - 7.3.1.11.X2 - Acked_Transitions Test for Normal to Fault Transitions		
	Test Conditionality	Must be executed if the IUT supports normal to fault event generating objects.
	Test Directives	
	Testing Hints	
…		

Specified Test Changes

[Remove Test 7.3.l.11.2]

~~7.3.1.11.2 Acked_Transitions Test for Latching Objects~~

~~Reason for Change: No test exists for this functionality.~~

~~Purpose: To verify that the Acked_Transitions property tracks the acknowledgment state for a transition type.~~

Test Concept: This test is a single transition test for latching life safety objects which are not able to perform the regular Acked_Transitions test for all transitions. An object, O1, in the IUT is made to generate a transition which requires an acknowledgement. The Acked_Transitions property is verified that the corresponding flag is cleared (set to FALSE). The transition is acknowledged, and the flag is verified to have been set back to TRUE.

Configuration Requirements: O1 is configured to generate events and to require acknowledgements for the transition being tested. O1 should have no event transitions which have outstanding acknowledgements.

Test Steps:

1. VERIFY Acked_Transitions = (TRUE, TRUE, TRUE)
2. MAKE (O1 transition)
3. WAIT (pTimeDelay)
4. BEFORE **Notification Fail Time**
   RECEIVE ConfirmedEventNotification Request,
   'Process Identifier' = (PI1: any valid process ID),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = O1,
   'Time Stamp' = (T1: any valid time stamp),
   'Notification Class' = (NC1: the class corresponding to the object being tested),
   'Priority' = (PRIO1: the value configured to correspond to the transition type),
   'Event Type' = (E1: any valid event type),
   'Message Text' = (optional, any valid message text),
   'Notify Type' = (the notify type configured for this event),
   'AckRequired' = TRUE,
   'From State' = S1,
   'To State' = S2,
   'Event Values' = (values appropriate to the event type)
5. TRANSMIT BACnet SimpleACK PDU
6. VERIFY pCurrentState = S2
7. IF S2 is NORMAL THEN
   VERIFY Acked_Transitions = (TRUE, TRUE, FALSE)
   VERIFY pStatusFlags = (FALSE, FALSE, ?, ?)
   ELSE IF S2 is FAULT THEN
   VERIFY Acked_Transitions = (TRUE, FALSE, TRUE)
   VERIFY pStatusFlags = (TRUE, TRUE, ?, ?)
   ELSE
   VERIFY Acked_Transitions = (FALSE, TRUE, TRUE)
   VERIFY pStatusFlags = (TRUE, FALSE, ?, ?)
8. TRANSMIT AcknowledgeAlarm Request,
   'Acknowledging Process Identifier' = PI1,
   'Event Object Identifier' = O1,
   'Event State Acknowledged' = S2,
   'Time Stamp' = T1,
   'Acknowledgement Source' = (a character string),
   'Time of Acknowledgment' = (the TD's current time)
9. RECEIVE BACnet SimpleACK PDU
10. IF (Protocol_Revision is present and Protocol_Revision ≥ 1) THEN
    BEFORE **Notification Fail Time**
    RECEIVE ConfirmedEventNotification Request,
    'Process Identifier' = PI1,
    'Initiating Device Identifier' = IUT,
    'Event Object Identifier' = O1,
    'Time Stamp' = The IUT's current time or sequence number,
    'Notification Class' = NC1,
    'Priority' = PRIO1,
    'Event Type' = E1,

~~'Message Text' = (optional, any valid message text),~~
~~'Notify Type' = ACK_NOTIFICATION,~~
~~'To State' = S2~~
~~ELSE~~
~~BEFORE **Notification Fail Time**~~
~~RECEIVE ConfirmedEventNotification-Request,~~
~~'Process Identifier' = PI1,~~
~~'Initiating Device Identifier' = IUT,~~
~~'Event Object Identifier' = O1,~~
~~'Time Stamp' = The IUT's current time or sequence number,~~
~~'Notification Class' = NC1,~~
~~'Priority' = PRIO1,~~
~~'Event Type' = E1,~~
~~'Message Text' = (optional, any valid message text),~~
~~'Notify Type' = ACK_NOTIFICATION~~
~~11. TRANSMIT BACnet-SimpleACK-PDU~~
~~12. VERIFY Acked_Transitions = (TRUE, TRUE, TRUE)~~

~~Notes to Tester: The UnconfirmedEventNotification service may be substituted for the ConfirmedEventNotification service, in which case the TD shall skip sending the BACnet-SimpleACK-PDU messages after receiving the notifications.~~

[Move test 7.3.1.11.1 from 135.1-2023 and modify as noted]

7.3.1.11.1 Acked_Transitions Test

Reason for Change: Corrected errata issues that are in 135.1-2023. Improved the text for Notes To Tester. Removed to_fault part of the test and moved to a new test.

Purpose: To verify that the Acked_Transitions property tracks whether or not an acknowledgment has been received for a previously issued event notification. It also verifies the interrelationship between Status_Flags and Event_State.

Test Concept: The IUT is configured such that the Event_Enable property indicates that all event transitions are to trigger an event notification. The Acked_Transitions property shall have the value (TRUE, TRUE, TRUE) indicating that all previous transitions have been acknowledged. Each event transition is triggered and the Acked_Transitions property is monitored to verify that the appropriate bit is cleared when a notification message is transmitted and reset if an acknowledgment is received.

Configuration Requirements: The Event_Enable and Acked_Transitions properties shall be configured with a value of (TRUE, TRUE, TRUE). For analog objects the Limit_Enable property shall be configured with the value (TRUE, TRUE). The referenced event-triggering property shall be set to a value that results in a NORMAL condition. The value of the Transitions parameter for all recipients shall be (TRUE, TRUE, TRUE).

Notes to Tester: The UnconfirmedEventNotification service may be substituted for the ConfirmedEventNotification service, in which case the TD shall skip sending the BACnet-SimpleACK-PDU messages after receiving the notifications.

Notes to Tester: For life safety objects that latch pMonitoredValue, the LifeSafetyOperation Service will be required to reset pMonitoredValue.

Test Steps:

1. VERIFY pCurrentState = NORMAL
2. VERIFY Acked_Transitions = (TRUE, TRUE, TRUE)
3. IF (Protocol_Revision is present AND Protocol_Revision >= 13) THEN
   VERIFY Status_Flags = (FALSE, FALSE, ?, ?)
4. IF (pMonitoredValue is writable) THEN
   WRITE pMonitoredValue = (a value that is OFFNORMAL)
   ELSE
   MAKE (pMonitoredValue have a value that is OFFNORMAL)
5. WAIT (pTimeDelay)
6. BEFORE **Notification Fail Time**

RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (PI1: any valid process ID),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the event-generating object configured for this test),
'Time Stamp' = (Toffnormal: any valid time stamp),
'Notification Class' = (the class corresponding to the object being tested),
'Priority' = (Poffnormal: the value configured to correspond to a TO-OFFNORMAL transition),
'Event Type' = (any valid event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = (the notify type configured for this event),
'AckRequired' = TRUE,
'From State' = NORMAL,
'To State' = OFFNORMAL,
'Event Values' = (values appropriate to the event type)

7. TRANSMIT BACnet-SimpleACK-PDU
8. VERIFY pCurrentState = OFFNORMAL
9. VERIFY Acked_Transitions = (FALSE, TRUE, TRUE)
10. IF (Protocol_revision is present AND Protocol_Revision >= 13 THEN
    VERIFY Status_Flags = (TRUE, FALSE, ?, ?)
11. IF (pMonitoredValue is writable) THEN
    WRITE pMonitoredValue = (a value that is NORMAL)
    ELSE
    MAKE (pMonitoredValue have a value that is NORMAL)
12. WAIT (pTimeDelayNormal)
13. BEFORE **Notification Fail Time**
    RECEIVE ConfirmedEventNotification-Request,
    'Process Identifier' = (PI2: any valid process ID),
    'Initiating Device Identifier' = IUT,
    'Event Object Identifier' = (the event-generating object configured for this test),
    'Time Stamp' = (Tnormal: any valid time stamp),
    'Notification Class' = (the class corresponding to the object being tested),
    'Priority' = (Pnormal: the value configured to correspond to a TO-NORMAL transition),
    'Event Type' = (any valid event type),
    'Message Text' = (optional, any valid message text),
    'Notify Type' = (the notify type configured for this event),
    'AckRequired' = TRUE,
    'From State' = OFFNORMAL,
    'To State' = NORMAL,
    'Event Values' = (values appropriate to the event type)
14. TRANSMIT BACnet-SimpleACK-PDU
15. VERIFY pCurrentState = NORMAL
16. VERIFY Acked_Transitions = (FALSE, TRUE, FALSE)
17. IF (Protocol_Revision is present AND Protocol_Revision >= 13) THEN
    VERIFY Status_Flags = (FALSE, FALSE, ?,?)
18. IF (the event-triggering object can be placed into a fault condition) THEN {
    MAKE (a condition exist that will cause the object to generate a fault condition)
    BEFORE **Notification Fail Time**
    RECEIVE ConfirmedEventNotification-Request,
    'Process Identifier' = (PI3: any valid process ID),
    'Initiating Device Identifier' = IUT,
    'Event Object Identifier' = (the event-generating object configured for this test),
    'Time Stamp' = (Tfault: any valid time stamp),
    'Notification Class' = (the class corresponding to the object being tested),
    'Priority' = (Pfault: the value configured to correspond to a TO-FAULT transition),
    'Event Type' = IF (Protocol_Revision < 13) THEN
    (any valid event type),
    ELSE
    CHANGE_OF_RELIABILITY,
    'Message Text' = (optional, any valid message text),

~~'Notify Type' = (the notify type configured for this event),~~
~~'AckRequired' = TRUE,~~
~~'From State' = NORMAL,~~
~~'To State' = FAULT,~~
~~'Event Values' = (values appropriate to the event type)~~
~~TRANSMIT BACnet-SimpleACK-PDU~~
~~VERIFY pCurrentState = FAULT~~
~~VERIFY Acked_Transitions = (FALSE, FALSE, FALSE)~~
~~TRANSMIT AcknowledgeAlarm-Request,~~
~~'Acknowledging Process Identifier' = (PI3),~~
~~'Event Object Identifier' = (the event-generating object configured for this test),~~
~~'Event State Acknowledged' = FAULT,~~
~~'Acknowledgement Source' = (a character string),~~
~~'Time Stamp' = (Tfault),~~
~~'Time of Acknowledgment' = (the TD's current time)~~
~~RECEIVE BACnet-SimpleACK-PDU~~
~~IF (Protocol_Revision is present AND Protocol_Revision ≥ 1) THEN~~
~~BEFORE **Notification Fail Time**~~
~~RECEIVE ConfirmedEventNotification-Request,~~
~~'Process Identifier' = (PI3),~~
~~'Initiating Device Identifier' = IUT,~~
~~'Event Object Identifier' = (the event-generating object configured for this test),~~
~~'Time Stamp' = (Tfault *the IUT's current time or sequence number*),~~
~~'Notification Class' = (the class corresponding to the object being tested),~~
~~'Priority' = (Pfault),~~
~~'Event Type' = IF (Protocol_Revision < 13)~~
~~(any valid event type),~~
~~ELSE~~
~~CHANGE_OF_RELIABILITY,~~
~~'Message Text' = (optional, any valid message text),~~
~~'Notify Type' = ACK_NOTIFICATION,~~
~~'To State' = FAULT~~
~~ELSE~~
~~BEFORE **Notification Fail Time**~~
~~RECEIVE ConfirmedEventNotification-Request,~~
~~'Process Identifier' = (PI3),~~
~~'Initiating Device Identifier' = IUT,~~
~~'Event Object Identifier' = (the event-generating object configured for this test),~~
~~'Time Stamp' = (Tfault *the IUT's current time or sequence number*),~~
~~'Notification Class' = (the class corresponding to the object being tested),~~
~~'Priority' = (Pfault),~~
~~'Event Type' = (any valid event type),~~
~~'Notify Type' = ACK_NOTIFICATION~~
~~TRANSMIT BACnet-SimpleACK-PDU~~
~~VERIFY Acked_Transitions = (FALSE, TRUE, FALSE)~~
~~}~~

18~~19~~. TRANSMIT AcknowledgeAlarm-Request,
'Acknowledging Process Identifier' = (PI2),
'Event Object Identifier' = (the event-generating object configured for this test),
'Event State Acknowledged' = NORMAL,
'Time Stamp' = (Tnormal),
'Acknowledgement Source' = (a character string),
'Time of Acknowledgment' = (the TD's current time)
19~~20~~. RECEIVE BACnet-SimpleACK-PDU
20~~21~~. IF (Protocol_Revision is present and Protocol_Revision ≥ 1) THEN
BEFORE **Notification Fail Time**
RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (PI2),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the event-generating object configured for this test),

'Time Stamp' = (~~Tnormal~~the IUT's current time or sequence number),
'Notification Class' = (the class corresponding to the object being tested),
'Priority' = (Pnormal),
'Event Type' = (any valid event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = ACK_NOTIFICATION,
'To State' = NORMAL

ELSE
BEFORE **Notification Fail Time**
RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (PI2),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the event-generating object configured for this test),
'Time Stamp' = (~~Tnormal~~the IUT's current time or sequence number),
'Notification Class' = (the class corresponding to the object bind tested),
'Priority' = (Pnormal),
'Event Type' = (any valid event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = ACK_NOTIFICATION

21~~22~~. TRANSMIT BACnet-SimpleACK-PDU
22~~23~~. VERIFY Acked_Transitions = (FALSE, TRUE, TRUE)
23~~24~~. TRANSMIT AcknowledgeAlarm-Request,
'Acknowledging Process Identifier' = (PI1),
'Event Object Identifier' = (the event-generating object configured for this test),
'Event State Acknowledged' = OFFNORMAL,
'Time Stamp' = (Toffnormal),
'Acknowledgement Source' = (a character string),
'Time of Acknowledgment' = (the TD's current time)
24~~25~~. RECEIVE BACnet-SimpleACK-PDU
25~~26~~. IF (Protocol_Revision is present and Protocol_Revision ≥ 1) THEN
BEFORE **Notification Fail Time**
RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (PI1),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the event-generating object configured for this test),
'Time Stamp' = (~~Toffnormal~~the IUT's current time or sequence number),
'Notification Class' = (the class corresponding to the object being tested),
'Priority' = (Poffnormal),
'Event Type' = (any valid event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = ACK_NOTIFICATION,
'To State' = OFFNORMAL

ELSE
BEFORE **Notification Fail Time**
RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (PI1),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the event-generating object configured for this test),
'Time Stamp' = (~~Toffnormal~~ the IUT;s current time or sequence number),
'Notification Class' = (the class corresponding to the object being tested),
'Priority' = (Poffnormal),
'Event Type' = (any valid event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = ACK_NOTIFICATION
26~~27~~. TRANSMIT BACnet-SimpleACK-PDU
27~~28~~. VERIFY Acked_Transitions = (TRUE, TRUE, TRUE)

[Add new test 7.3.1.11.X2]

7.3.1.11.X2 Acked_Transitions Test for Normal to Fault Transitions

Reason for Change: New test.

Purpose: To verify that the Acked_Transitions property tracks whether or not an acknowledgment has been received for a previously issued fault event notification. It also verifies the interrelationship between Status_Flags and Event_State.

Test Concept: The IUT is configured such that the Event_Enable property indicates that fault event transitions are to trigger an event notification. The Acked_Transitions property shall have the value (?, TRUE, ?). The fault event transition is triggered and the Acked_Transitions property is monitored to verify that the appropriate bit is cleared when a notification message is transmitted and reset when an acknowledgment is received.

Configuration Requirements: The Event_Enable and Acked_Transitions properties shall be configured with a value of (?, TRUE, ?). The referenced event-triggering property shall be set to a value that results in a NORMAL condition. The value of the Transitions parameter for all recipients shall be (?, TRUE, ?).

Notes to Tester: The UnconfirmedEventNotification service may be substituted for the ConfirmedEventNotification service, in which case the TD shall skip sending the BACnet-SimpleACK-PDU messages after receiving the notifications.

Test Steps:

1. VERIFY pCurrentState = NORMAL
2. VERIFY Acked_Transitions = (?, TRUE, ?)
3. IF (Protocol_Revision is present AND Protocol_Revision >= 13) THEN
   VERIFY Status_Flags = (FALSE, FALSE, ?, ?)
4. MAKE (a condition exist that will cause the object to generate a fault condition)
5. BEFORE **Notification Fail Time**
   RECEIVE ConfirmedEventNotification-Request,
   'Process Identifier' = (PI3: any valid process ID),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (the event-generating object configured for this test),
   'Time Stamp' = (Tfault: any valid time stamp),
   'Notification Class' = (the class corresponding to the object being tested),
   'Priority' = (Pfault: the value configured to correspond to a TO-FAULT transition),
   'Event Type' = IF (Protocol_Revision < 13) THEN
   (any valid event type),
   ELSE
   CHANGE_OF_RELIABILITY,
   'Message Text' = (optional, any valid message text),
   'Notify Type' = (the notify type configured for this event),
   'AckRequired' = TRUE,
   'From State' = NORMAL,
   'To State' = FAULT,
   'Event Values' = (values appropriate to the event type)
6. TRANSMIT BACnet-SimpleACK-PDU
7. VERIFY pCurrentState = FAULT
8. VERIFY Acked_Transitions = (?, FALSE, ?)
9. IF (Protocol_revision is present AND Protocol_Revision >= 13 THEN
   VERIFY Status_Flags = (FALSE, TRUE, ?, ?)
10. TRANSMIT AcknowledgeAlarm-Request,
    'Acknowledging Process Identifier' = (PI3),
    'Event Object Identifier' = (the event-generating object configured for this test),
    'Event State Acknowledged' = FAULT,
    'Acknowledgement Source' = (a character string),
    'Time Stamp' = (Tfault),
    'Time of Acknowledgment' = (the TD’s current time)
11. RECEIVE BACnet-SimpleACK-PDU
12. IF (Protocol_Revision is present AND Protocol_Revision ≥ 1) THEN
    BEFORE **Notification Fail Time**
    RECEIVE ConfirmedEventNotification-Request,
    'Process Identifier' = (PI3),
    'Initiating Device Identifier' = IUT,

'Event Object Identifier' = (the event-generating object configured for this test),
'Time Stamp' = (Tfault the IUT's current time or sequence number),
'Notification Class' = (the class corresponding to the object being tested),
'Priority' = (Pfault),
'Event Type' = IF (Protocol_Revision < 13)
(any valid event type),
ELSE
CHANGE_OF_RELIABILITY,
'Message Text' = (optional, any valid message text),
'Notify Type' = ACK_NOTIFICATION,
'To State' = FAULT

ELSE

BEFORE **Notification Fail Time**

RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (PI3),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the event-generating object configured for this test),
'Time Stamp' = (Tfault the IUT's current time or sequence number),
'Notification Class' = (the class corresponding to the object being tested),
'Priority' = (Pfault),
'Event Type' = (any valid event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = ACK_NOTIFICATION

13. TRANSMIT BACnet-SimpleACK-PDU
14. VERIFY Acked_Transitions = (?, TRUE, ?)