BACnet® TESTING LABORATORIES
ADDENDA

Addendum cr2 to BTL Test Package 20.0.1

Revision v2
Revised 10/21/2022

Approved by the BTL Working Group on 2022-10-03.
Approved by the BTL Working Group Voting Members on 2022-11-07.
Published on 2022-11-10.

BACnet is a registered trademark of ASHRAE. ASHRAE does not endorse, approve or test products for compliance with ASHRAE standards. Compliance of listed products to the requirements of ASHRAE Standard 135 is the responsibility of BACnet International. BTL is a registered trademark of BACnet International.

[This foreword and the "Overview" on the following pages are not part of this Test Package. They are merely informative and do not contain requirements necessary for conformance to the Test Package.]

FOREWORD

The purpose of this addendum is to present current changes being made to the BTL Test Package. These modifications are the result of change proposals made pursuant to the continuous maintenance procedures and of deliberations within the BTL-WG Committee. The changes are summarized below.

In the following document, language to be added to existing clauses within the BTL Test Package 20.0.1 is indicated through the use of *italics*, while deletions are indicated by ~~strikethrough~~. Where entirely new subclauses are proposed to be added, plain type is used throughout

In contrast, changes to BTL Specified Tests also contain a yellow highlight to indicate the changes made by this addendum. When this addendum is applied, all highlighting will be removed. Change markings on tests will remain to indicate the difference between the new test and an existing 135.1 test. If a test being modified has never existed in 135.1, the applied result should not contain any change markings. When this is the case, square brackets will be used to describe the changes required for this test.

Each addendum can stand independently unless specifically noted via dependency within the addendum. If multiple addenda change the same test or section, each future released addendum that changes the same test or section will note in square brackets whether or not those changes are reflected.

BTL-20.0.1 cr2-1: Add Test Coverage for Extended MS/TP Frames [BTLWG-1058, CR-0560]

Overview:

BTLWG-560 added test coverage for extended MS/TP frames for MS/TP Master Nodes but did not add similar testing for MS/TP slave devices. The omission for slaves was intentional as the appropriate DNER frame type, 33, was only defined for master nodes. However, per IC-135-2020-3, it was determined that the original language was incorrect and slave nodes are allowed to use extended frames.

This work item also adds a test applicable to both masters and slaves to ensure they will ignore unknown frame types.

Changes:

Checklist Changes

[In BTL Checklist, add entry in Section 9 Data Link Layer]

Data Link Layer - MS/TP - Slave Node		
	R	Base Requirements
	O	Supports configuration through Network Port object
	O[1]	Supports Extended MS/TP Frames (over 501 octets)
[1]Protocol_Revision 16 or higher must be claimed		

Test Plan Changes

[In BTL Test Plan, add entries for Supports Extended Frames (over 501 octets) in section 9]

9.2.3 Supports Extended MS/TP Frames (over 501 octets)

The IUT can transmit and receive messages with an NPDU > 501 octets

BTL - 12.1.3.X1 - Frame Type Based on Transmitted NPDU Size		
	Test Conditionality	Must be executed
	Test Directives	Execute the test such that the transmitted NPDU sizes are near the 501 octet boundary.
	Testing Hints	

[In BTL Test Plan, add a test to the Base Requirements for MSTP Master and Slave devices]

9.1.1 Base Requirements

Base requirements for all MS/TP master devices.

…		
BTL - 12.1.3.X - Ignores Unknown Frame Types		
	Test Conditionality	If the IUT supports extended frames, this test shall be skipped.
	Test Directives	
	Testing Hints	

9.2.1 Base Requirements

Base requirements for all MS/TP slave devices.

…
BTL - 12.1.3.X - Ignores Unknown Frame Types

	Test Conditionality	If the IUT supports extended frames, this test shall be skipped.
	Test Directives	
	Testing Hints	

Test Changes

[In BTL Specified Tests, add new test 12.1.3.X]

12.1.3.X Ignores Unknown Frame Types

Reason for Change: No test exists for this functionality.

Purpose: To verify that the IUT will quietly ignore unknown frame types.

Test Concept: The TD sends MSTP frames to the IUT with extended frame types (32 and 33). The IUT is observed to verify that it quietly ignores the unknown frame types and does not reset.

Test Configuration: None

Test Steps:

```
1.  VERIFY System_Status = OPERATIONAL | OPERATIONAL_READ_ONLY
2.  TRANSMIT (any BACnet service choice, NPDU > 501 octets)
            Frame Type = 32 -- DER frame
3.  CHECK (verify that the IUT does not send a frame in response and does not reset)
4.  VERIFY System_Status = OPERATIONAL | OPERATIONAL_READ_ONLY
5.  TRANSMIT (any BACnet service choice, NPDU > 501 octets)
            Frame Type = 33 -- DNER frame
6.  CHECK (verify that the IUT does not send a frame in response and does not reset)
7.  VERIFY System_Status = OPERATIONAL | OPERATIONAL_READ_ONLY
```

BTL-20.0.1 cr2-2: Fix Backup and Restore Tests for Record Access [BTLWG-1090, CR-0352]

Overview:

Per the SSPC ruling, no changes were made for this CR.

Changes:

Checklist Changes

None

Test Plan Changes

None

Test Changes

None

BTL-20.0.1 cr2-3: Value Source Test Corrections [BTLWG-1111, CR-0499]

Overview:

Per BTL-CR-0499, tests 7.3.1.6.11 and 7.3.1.6.12 are corrected.

Changes:

Checklist Changes

None

Test Plan Changes

None

Test Changes

[In BTL Specified Tests, modify existing tests, as shown]

7.3.1.6.11 Minimum_Off_Time - Value Source Mechanism
Reason for Change: This test is not specified in any SSPC proposal.

Purpose: To verify that the value source used for priority 6 is the commanded object while Minimum_Off_Time is in effect.

Test Concept: A commandable object which supports the value source mechanism is selected for the test. When Minimum_Off_Time takes effect, the Present_Value is written. The Value_Source and Value_Source_Array properties are monitored to verify that the source for priority 6 is the commanded object.

Configuration Requirements: The object, O1, to be tested shall be configured such that slot 6 in the Priority_Array ~~and Value_Source_Array~~ has a value of NULL. The object being tested must also be configured with Minimum_Off_Time values sufficiently large enough to allow execution of this test. If no object exists with Minimum_Off_Time property, this test shall be skipped.

Notes to Tester: Comparisons with O1 are with or without the optional device-identifier of IUT.

Test Steps:

1. VERIFY Value_Source = (any valid value)
2. VERIFY Priority_ Array = NULL, ARRAY INDEX = 6
~~3. VERIFY Value_Source_Array = NULL, ARRAY INDEX = 6~~
~~4~~3. WRITE Present_Value = INACTIVE, PRIORITY > 6
~~5~~4. VERIFY Present_Value = INACTIVE
~~6~~5. VERIFY Priority_Array = INACTIVE, ARRAY INDEX = 6
~~7~~6. VERIFY Value_Source = O1
~~8~~7. VERIFY Value_Source_Array = ~~O1~~ Value_Source, ARRAY INDEX = 6
~~9~~8. WAIT (Minimum ON/OFF Fail Time + Minimum_Off_Time)
~~10~~9. VERIFY Value_Source ~~= 'None'~~ <> O1

7.3.1.6.12 Minimum_On_Time - Value Source Mechanism
Reason for Change: This test is not specified in any SSPC proposal.

Purpose: To verify that the value source used for priority 6 is the commanded object while Minimum_On_Time is in effect.

Test Concept: A commandable object which supports the value source mechanism is selected for the test. When Minimum_On_Time takes effect, the Present_Value is written. The Value_Source and Value_Source_Array properties are monitored to verify that the source for priority 6 is the commanded object.

Configuration Requirements: The object, O1, to be tested shall be configured such that slot 6 in the Priority_Array ~~and Value_Source_Array~~ has a value of NULL. The object being tested must also be configured with Minimum_On_Time values sufficiently large enough to allow execution of this test. If no object exists with Minimum_On_Time property, this test shall be skipped.

Notes to Tester: Comparisons with O1 are with or without the optional device-identifier of IUT.

Test Steps:

1. VERIFY Value_Source = (any valid value)
2. VERIFY Priority_ Array = NULL, ARRAY INDEX = 6
~~3. VERIFY Value_Source_Array = NULL, ARRAY INDEX = 6~~
~~4~~3. WRITE Present_Value = ACTIVE, PRIORITY > 6
~~5~~4. VERIFY Present_Value = ACTIVE
~~6~~5. VERIFY Priority_Array = ACTIVE, ARRAY INDEX = 6
~~7~~6. VERIFY Value_Source = O1
~~8~~7. VERIFY Value_Source_Array = ~~O1~~ Value_Source, ARRAY INDEX = 6
~~9~~8. WAIT (Minimum ON/OFF Fail Time + Minimum_On_Time)
~~10~~9. VERIFY Value_Source ~~= 'None'~~ <> O1

BTL-20.0.1 cr2-4: Network Number Is Changes for Virtual Devices [BTLWG-1198]

Overview:

Virtual routers have some differences from routers to external devices. Test 10.2.X1 ensures that routers send out Network-Number-Is on all ports but it should not apply to virtual ports

Changes:

Checklist Changes

None

Test Plan Changes

10.1 Network Management - Routing

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.

…

BTL - 10.2.X1 - Initiates Network-Number-Is on Startup		
	Test Conditionality	If the IUT supports Protocol_Revision 11 or greater, this test must be executed.
	Test Directives	Since no traffic at virtual ports is ever observed, it is not necessary for Network-Number-Is packets to be formed or emitted on networks which are only connected to virtual ports.
	Testing Hints	

Test Changes

None

BTL-20.0.1 cr2-5: Update Testing for Unspecified Day Of Week [BTLWG-1215, CR-0531]

Overview:

Based on BTL-CR-0531, several tests need to be updated to test that writing unspecified day of week is properly tested.

Note, tests 7.2.X3, 7.2.X4, 9.23.2.X9, and 9.23.2.X12 were also reviewed, and no changes are required.

Changes:

Checklist Changes

None

Test Plan Changes

3.8.1 Base Requirements

Base requirements must be met by any IUT that can contain Calendar Objects

BTL - 7.3.2.8.1 - Single Date Rollover Test		
	Test Conditionality	If the IUT does not contain any calendars with a writable Date_List~~ and does not contain any calendars which can be configured with a Date entry~~, this test shall be skipped.
	Test Directives	
	Testing Hints	
BTL - 7.3.2.8.2 - Date Range Test		
	Test Conditionality	If the IUT does not contain any calendars with a writable Date_List~~ and dos not contain any calendars which can be configured with a BACnetDateRange entry~~, this test shall be skipped.
	Test Directives	
	Testing Hints	
BTL - 7.3.2.8.3 - WeekNDay Test		
	Test Conditionality	If the IUT does not contain any calendars with a writable Date_List~~ and does not contain any calendars which can be configured with a BACnetWeekNDay entry~~, this test shall be skipped.
	Test Directives	
	Testing Hints	
BTL - 7.2.X1 - Date Pattern Properties Test		
	Test Conditionality	If the IUT does not contain any calendars with a writable Date_List~~ and does not contain any calendars which can be configured with a BACnetCalendarEntry entry containing a Date~~, this test shall be skipped.
	Test Directives	Apply to the Date_List property.
	Testing Hints	
BTL - 7.2.X7 - BACnetDateRange Non-Pattern Properties Test		
	Test Conditionality	This test shall only be applied to devices claiming Protocol_Revision 11 or higher. If the IUT does not contain any calendars with a writable Date_List~~ and does not contain any calendars which can be configured with a BACnetCalendarEntry entry containing a BACnetDateRange~~, this test shall be skipped.
	Test Directives	Apply to Date_List property.
	Testing Hints	
BTL - 7.2.X8 - BACnetDateRange Open-Ended Pattern Properties Test		
	Test Conditionality	This test shall only be applied to devices claiming Protocol_Revision 11 or higher.

		If the IUT does not contain any calendars with a writable Date_List~~ and does not contain any calendars which can be configured with a BACnetCalendarEntry entry containing a BACnetDateRange~~, this test shall be skipped.
	Test Directives	Apply to Date_List property.
	Testing Hints	
BTL - 9.23.2.X12 - BACnetDateRange Non-Pattern Properties Test using WritePropertyMultiple Service		
	Test Conditionality	This test shall only be applied to devices claiming Protocol_Revision 11 or higher and which supports execution of WritePropertyMultiple. If the IUT does not contain any calendars with a writable Date_List~~ and does not contain any calendars which can be configured with a BACnetCalendarEntry entry containing a BACnetDateRange~~, this test shall be skipped.
	Test Directives	Apply to the Date_List property.
	Testing Hints	

3.37.1 Base Requirements

Base requirements must be met by any IUT that can contain Accumulator objects.

…		
~~**BTL - 7.2.X6 - DateTime Non-Pattern Properties Test**~~		
	~~**Test Conditionality**~~	~~The test shall be skipped if the IUT claims Protocol_Revision 9 or prior, or if the IUT does not support Accumulator objects with writable Value_Change_Time properties.~~
	~~**Test Directives**~~	~~Apply to the Value_Change_Time property, if writable.~~
	~~**Testing Hints**~~	

Test Changes

[Note these tests do not exist in 135.1 and therefore no italics are used. The strike-through is for clarity on what changed to the test from the previous version.]

[Modify test 7.2.X1 BTL Specified Tests]

7.2.X1 Date Pattern Properties Test

Reason for Change: Addendum 135-2001a-1 adds odd and even month support, and last-day-of-the-month special value. Addendum 135-2008h-8 adds odd and even day support. Addendum 135-2008ac-1 clarifies when wildcards are allowed in dates and times. Test does not exist in 135.1-2013. Allow for non-configurable Date_List properties.

Purpose: To verify that the property being tested accepts special date field values.

Test Concept: The property being tested, P1, is written with each of the special date field values to ensure that the property accepts them. A date, D1, is selected which is within the date range that the IUT will accept for the property. The value, 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. The list of Specials comes from the Chapter 21 Application Types section on Date. The day-of-week field is redundant information and can be regenerated from the other fields. In order to not fail devices which always ensure this field is consistent with the other fields in the date value, the testing of ~~use of an~~ unspecified day of week is tested separately and allows either the device to return the unspecified day of week or the correctly calculated day of week~~always tested in conjunction with another pattern value~~.

~~Configuration Requirements: The IUT shall be configured with a Calendar object that contains a Date_List with a single BACnetCalendarEntry in the form of a Date. If Date_List property cannot be configured with a BACnetCalendarEntry in the form of a Date, then this test shall be skipped.~~

Notes to Tester: if P1 is an array, then an array index shall be provided in the WRITE and VERIFY operations.

Test Steps:

```
1.  IF (Protocol_Revision is not present or Protocol_Revision < 4) THEN
            Specials = (year unspecified, month unspecified, day of month unspecified)
    ELSE IF (Protocol_Revision >= 4 and Protocol_Revision < 10) THEN
            Specials = (year unspecified, month unspecified, day of month unspecified,
            odd months, even months, last day of month)
     ELSE
            Specials = (year unspecified, month unspecified, day of month unspecified,
            odd months, even months, last day of month, even days, odd days)
2.  REPEAT SV = (each value in Specials) DO {
            ~~IF (SV <> day of week unspecified) THEN~~
                    ~~V1 = D1 updated with the value SV~~
            ~~ELSE~~
                    ~~V1 = D1 updated with the value SV and any other value from Specials~~
            ~~WRITE P1 = (V1)~~
            ~~VERIFY P1 = (V1)~~
            WRITE P1 = (D1 updated with the value SV)
            VERIFY P1 = (D1 updated with the value SV)
            }
3.  WRITE P1 = (D1 with day of week unspecified)
4.  VERIFY P1 = (D1 with day of week unspecified or day of week containing the correct calculated value)
```

[Modify test 7.2.X6 BTL Specified Tests]

7.2.X6 DateTime Non-Pattern Properties Test

Reason for Change: Addendum 135-2001a-1 adds odd and even month support, and last-day-of-the-month special value. Addendum 135-2008*h*-8 adds odd and even day support. Addendum 135-2008*ac*-1 clarifies when wildcards are allowed in dates and times. Test does not exist in 135.1-2013.

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

Test Concept: 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 an array index shall be provided in the TRANSMIT portion of step 1.

Test Steps:

```
1.  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 {
            TRANSMIT WriteProperty-Request
                    'Object Identifier' =       O1,
                    'Property Identifier' =     P1,
                    'Property Value' =          (DT1 updated with the special value SV)
            RECEIVE BACnet-Error-PDU
                    'Error Class' =             PROPERTY,
                    'Error Code' =              VALUE_OUT_OF_RANGE
```

[Modify test 7.2.X7 BTL Specified Tests]

7.2.X7 BACnetDateRange Non-Pattern Properties Test

Reason for Change: Addendum 135-2008*ac*-1 clarifies in the clause 12 preamble, when wildcards are allowed in BACnetDateRange. Allow for non-configurable Date_List properties.

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

Test Concept: A BACnetDateRange property, or property that is a complex datatype containing a BACnetDateRange, P_1 is written with each of the special date field values to ensure that the property does not accept them. Each half of the dateRange DR_1 is selected so it is within the range that the IUT will accept for the property. The value, V_1, written to the property is the dateRange DR_1 with one of its fields replaced with one of the date special values. If the property is a complex datatype such as a BACnetCalenderEntry, the other fields in the value shall be set within the range accepted by the IUT.

Configuration Requirements: This test shall only be applied to devices claiming Protocol_Revision 11 or higher. The IUT shall be configured with a Calendar object that contains a Date_List with a single BACnetCalendarEntry in the form of a BACnetDateRange. If Date_List property cannot be configured with a BACnetCalendarEntry in the form of a BACnetDateRange, then this test shall be skipped.

Notes to Tester: if P_1 is an array, then an array index shall be provided in the TRANSMIT portion of step 1.

Test Steps:

```
1.  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 {
            TRANSMIT WriteProperty-Request
                    'Object Identifier' =       O1,
                    'Property Identifier' =     P1,
                    'Property Value' =          (DR1 with startDate updated with the special value SV)
            RECEIVE BACnet-Error-PDU
                    'Error Class' =             PROPERTY,
                    'Error Code' =              VALUE_OUT_OF_RANGE
            TRANSMIT WriteProperty-Request
                    'Object Identifier' =       O1,
                    'Property Identifier' =     P1,
                    'Property Value' =          (DR1 with endDate updated with the special value SV)
            Receive BACnet-Error-PDU
                    'Error Class' =             PROPERTY,
                    'Error Code' =              VALUE_OUT_OF_RANGE
```

[Modify test 9.23.2.X11 BTL Specified Tests]

9.23.2.X11 DateTime Non-Pattern Properties Test using WritePropertyMultiple Service

Reason for Change: No test exists for this functionality. This test is not in any SSPC proposal.

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

Test Concept: 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:

```
1.  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 {
2.      TRANSMIT WritePropertyMultiple-Request,
            'Object Identifier' = O1,
            'Property Identifier' = P1,
            'Property Value' = (DT_1 updated with the special value SV)
3.      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-20.0.1 cr2-6: Correct Test for Reliability_Evaluation_Inhibit [BTLWG-1235]

Overview:

Reliability_Evaluation_Inhibit testing must be done with Out_Of_Service == FALSE.

Changes:

Checklist Changes

None

Test Plan Changes

None

Test Changes

[Modify test 7.3.1.21.1 in BTL Specified Tests]
[Fixed incorrect Reason For Change]

7.3.1.21.1 Reliability_Evaluation_Inhibit Test

Reason for Change: Changed to allow for non-writable Reliability_Evaluation_Inhibit property. Updated to ensure that the object is in service during the test.

Purpose: To verify that Reliability_Evaluation_Inhibit controls whether or not fault conditions are detected.

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*, if event reporting is supported,* that a notification is generated. Set Reliability_Evaluation_Inhibit to TRUE. Verify that the Reliability changes to NO_FAULT_DETECTED and*, if event reporting is supported,* 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.

Test Configuration: O1 is configured to detect and*, if event reporting is supported,* report unconfirmed events, is in the NORMAL state, *has Out_Of_Service set to FALSE* 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 *and which meets these configuration requirements* then this test shall be skipped.

Test Steps:
[Test Steps need to be renumberd when this test is applied in 135.1, the changes are not noted here for clarity.]

```
1.  VERIFY Out_Of_Service = FALSE
2.  VERIFY pCurrentState = NORMAL
3.  VERIFY Reliability = NO_FAULT_DETECTED
4.  MAKE(~~a condition exist that would cause O1 to generate a TO_FAULT transition~~a fault condition exist for O1)
5.  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)
6.  VERIFY Reliability <> NO_FAULT_DETECTED
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 condition exist that would cause O1 to generate a TO_NORMAL transition~~a fault condition exist for O1)
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)
```

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

BTL-20.0.1 cr2-7: Correct Numerical Algorithm Tests [BTLWG-1256, CR-0031]

Overview:

Jira item BTLWG-1256. From CR-0031, test 8.4.3.1 should not state it is applicable to numeric datatypes when the algorithm is restricted to REALs.

Changes:

Checklist Changes

None

Test Plan Changes

[change all references in BTL Test Package to BTL - 8.4.3.1 - Numerical Algorithm (ConfirmedEventNotification)]
[change all references in BTL Test Package to BTL - 8.54.3.1 - Numerical Algorithm (UnconfirmedEventNotification)]

Test Changes

[Modify 8.4.3.1 and move into BTL Specified Tests]

8.4.3.1 Numerical Algorithm (ConfirmedEventNotification)

Reason For Change: The alg is only applicable to REAL data types so it should not use the term numeric.

Purpose: To verify the correct operation of the CHANGE_OF_VALUE event algorithm as applied to *REAL* ~~numerical~~ datatypes.

Test Concept: The object begins the test in a NORMAL state. pMonitoredValue is changed by a value that is less than pIncrement. The tester verifies that no event notification is transmitted. pMonitoredValue is changed again to a value that differs from the original value by an amount greater than pIncrement. The tester verifies that an event notification message is transmitted and that the proper event state transitions occur.

Configuration Requirements: The IUT shall be configured such that the Event_Enable property has a value of TRUE for the TO-NORMAL transition. The 'Issue_Confirmed_Notifications' parameter shall have a value of TRUE. The event-generating object shall be in a NORMAL state at the start of the test.

Test Steps:

…

[Modify 8.5.3.1 and move into BTL Specified Tests]

8.5.3.1 Numerical Algorithm (UnconfirmedEventNotification)

Purpose: To verify the correct operation of the CHANGE_OF_VALUE event algorithm applied to ~~Integer or~~ Real datatypes.

Test Concept: The test concept corresponds to 8.4.3.1.

Configuration Requirements: The configuration requirements are identical to those in 8.4.3.1, except that the 'Issue Confirmed Notifications' parameter shall have a value of FALSE.

Test Steps: The test steps for this test case are identical to the test steps in 8.4.3.1 except that the ConfirmedEventNotification requests are UnconfirmedEventNotification requests and the TD does not acknowledge receiving the notifications.

Notes to Tester: The passing results for this test case are identical to the ones in 8.4.3.1 except that the event notifications shall be conveyed using an UnconfirmedEventNotification service request. The MAC address used for these messages shall be either a broadcast that reaches the local network of the TD or the MAC address of the TD.

BTL-20.0.1 cr2-8: Correct Out_Of_Service Test for Optional Reliability Property [BTLWG-1288]

Overview:

Jira item BTLWG-288. Test 7.3.1.1.X1 assumes that Reliability property exists in the object. The test needs to be modified for object instances which do not have a Reliability property.

Changes:

Checklist Changes

None

Test Plan Changes

None

Test Changes

[Modify 7.3.1.1.X1 from BTL Specified Tests]

7.3.1.1.X1 Out_Of_Service, Status_Flags, and Reliability Test

~~**7.3.1.1 Out_Of_Service, Status_Flags, and Reliability Test**~~

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

~~BACnet Reference Clauses: 12.1.7, 12.1.9, 12.1.10, 12.2.7, 12.2.9, 12.2.10, 12.3.7, 12.3.9, 12.3.10, 12.4.6, 12.4.8,12.4.9, 12.6.7, 12.6.9, 12.6.10, 12.7.7, 12.7.9, 12.7.10, 12.8.6, 12.8.8, 12.8.9, 12.15.8, 12.15.10, 12.15.11, 12.16.8, 12.16.10, 12.16.11, 12.17.6, 12.17.8, 12.17.9, 12.18.7, 12.18.9, 12.18.10, 12.19.7, 12.19.9, 12.19.10, 12.20.6, 12.20.8, 12.20.9, 12.23.7, 12.23.9, and 12.23.10.~~

Purpose: ~~This test case verifies that~~ *To verify that* Present_Value is writable when Out_Of_Service is TRUE and~~. It also~~ *that* the interrelationship between the Out_Of_Service, Status_Flags, and Reliability properties. ~~If the PICS indicates that the Out_Of_Service property of the object under test is not writable, and if the value of the property cannot be changed by other means, then this test shall be omitted. This test applies to Accumulator, Analog Input, Analog Output, Analog Value, Binary Input, Binary Output, Binary Value, Life Safety Point, Life Safety Zone, Multi-state Input, Multi-state Output, Multi-state Value, Loop and Pulse Converter objects.~~

Test Concept: ~~The IUT will select one instance of each appropriate object type and test it as described. If the Reliability property is not supported then step 4 shall be omitted.~~ *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.  IF Reliability is present THEN
        READ R1 = Reliability
3~~1~~. IF (Out_Of_Service is writable) THEN
        WRITE Out_Of_Service = TRUE
    ELSE
```

```
          MAKE (Out_Of_Service TRUE)
4~~2~~.  VERIFY Out_Of_Service = TRUE
5~~3~~.  VERIFY Status_Flags = (?, ~~FALSE~~?, ?, TRUE)
6~~4~~.  REPEAT X = (all values meeting the functional range requirements of 7.2.1) DO {
          WRITE Present_Value, PTY1 = X
          VERIFY Present_Value = X
          }
7~~5~~.  IF (Reliability is present and writable) THEN
          REPEAT X = (all values of the Reliability enumeration appropriate to the object type except
                      NO_FAULT_DETECTED) DO {
              WRITE Reliability = X
              VERIFY Reliability = X
              VERIFY Status_Flags = (?, TRUE, ?, TRUE)
              WRITE Reliability = NO_FAULT_DETECTED
              VERIFY Reliability = NO_FAULT_DETECTED
              VERIFY Status_Flags = (?, FALSE, ?, TRUE)
              }
8~~6~~.  IF (Out_Of_Service is writable) THEN
          WRITE Out_Of_Service = FALSE
      ELSE
          MAKE (Out_Of_Service FALSE)
9~~7~~.  VERIFY Out_Of_Service = FALSE
10~~8~~.VERIFY Status_Flags = ~~(?, ?, ?, FALSE)~~SF1
11.   IF Reliability is present THEN
          VERIFY Reliability = R1
```

~~Notes to Tester: If the object being tested is commandable and there is an internal process writing to the Present_Value property, then each WriteProperty request shall contain a priority sufficient to override the internal process. After step 4 the priority array slot shall be relinquished.~~

BTL-20.0.1 cr2-9: Clarify Out_Of_Service Testing [BTLWG-1294]

Overview:

Addendum 135-2016bl-3. Clarify Out_Of_Service

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

BTL Test Plan also added a new test case BTL - 7.3.1.1.X2 - Out_Of_Service for Commandable Value Objects Test. The test conditionality of this test case is "If the object is commandable, this test must be executed." However, if the device does not support modifying Present_Value via software local to the device, this test should be skipped. Same language is proposed through this proposal.

Changes:

Checklist Changes

None

Test Plan Changes

[Replace Test Conditionality language with the following language in the **BTL Test Plan-20.0_Final**:
Clause 3.3.2, Analog Value object type, p. 14,
Clause 3.7.2, Binary Value object type, p. 23,
Clause 3.16.2, Multi-state Value object type, p. 39,
Clause 3.25.2, CharacterString Value object type, p. 52,
Clause 3.29.2, DateTime Value object type, p. 60,
Clause 3.31.2, Large Analog Value object type, p. 64,
Clause 3.24.2, BitString Value object type, p. 50,
Clause 3.32.2, OctetString Value object type, p. 66,
Clause 3.35.2, Time Value object type, p. 72,
Clause 3.30.2, Integer Value object type, p. 62,
Clause 3.33.2, Positive Integer Value object type, p. 68,
Clause 3.27.2, Date Value object type, p. 56,
Clause 3.28.2, DateTime Pattern Value object type, p. 58,
Clause 3.34.2, Time Pattern Value object type, p. 70,
Clause 3.26.2, Date Pattern Value object type, p. 54]

3.X.2 Supports Writable Out_Of_Service Property
…

BTL - 7.3.1.1.X2 - Out_Of_Service for Commandable Value Objects Test		
	Test Conditionality	~~If the object is commandable, this test must be executed.~~ This test shall be skipped if the device does not contain a commandable instance of this object type which supports modification of Present_Value via software local to the device.
	Test Directives	
	Testing Hints	

Test Changes

None

BTL-20.0.1 cr2-10: Fix the Blink-Warn Command Tests [BTLWG-1307, CR-0533]

Overview:

Per CR-0533 test 7.3.1.X41.Y3 is incorrect. CRR-0533 specifies the change for this test and 7.3.1.X41.Y2.

Changes:

Checklist Changes

None

Test Plan Changes

None

Test Changes

[Modify tests as shown. Note these tests do not exist in 135.1 and therefore no italics are shown. Strike-out is shown for clarity. All changed lines have been highlighted.]

7.3.1.X41.Y2 Blink-Warn WARN_OFF Command Test

…

Test Steps:

```
1.  VERIFY Priority_ Array = V1, ARRAY INDEX = PTY1
2.  VERIFY Blink_Warn_Enable = TRUE
3.  VERIFY Egress_Time > 0
4.  VERIFY Egress_Active = FALSE
5.  WRITE PROP_REF = C1, PRIORITY = PTY1
6.  T1 = current local time
~~7.  BEFORE Internal Processing Fail Time~~
~~        CHECK (blink-warn occurred)~~
7. VERIFY Priority_ Array = V1, ARRAY INDEX = PTY1
8.  WHILE (Egress_Active = TRUE)
      WAIT 1s
        ~~VERIFY Priority_ Array = V1, ARRAY INDEX = PTY1~~
9.  T2 = current local time
10. CHECK (blink warn occurred)
11. VERIFY Egress_Time ~= (T2 - T1) ~~(T1 – T2) +/- Internal Processing Fail Time~~
12. VERIFY Priority_ Array = V2, ARRAY INDEX = PTY1
```

7.3.1.X41.Y3 Blink-Warn WARN_RELINQUISH Command Test

…

Test Steps:

```
1.  VERIFY Priority_ Array = V1, ARRAY INDEX = PTY1
2.  VERIFY Blink_Warn_Enable = TRUE
3.  VERIFY Egress_Time > 0
4.  VERIFY Egress_Active = FALSE
5.  WRITE PROP_REF = C1, PRIORITY = PTY1
6.  T1 = current local time
```

```
~~7. BEFORE **Internal Processing Fail Time**~~
~~        CHECK (blink-warn occurred)~~
7.  VERIFY Priority_ Array = V1, ARRAY INDEX = PTY1
8.  WHILE (Egress_Active = TRUE)
        WAIT 1s
        ~~VERIFY Priority_ Array = V1, ARRAY INDEX = PTY1~~
9.  T2 = current local time
10. CHECK(blink warn occurred)
11. VERIFY Egress_Time ~= (T2 - T1) ~~(T1 - T2) +/- **Internal Processing Fail Time**~~
12  VERIFY Priority_ Array = NULL, ARRAY INDEX = PTY1
```

BTL-20.0.1 cr2-11: Fix Conditionality of MS/TP Max_Master Testing [BTLWG-1308, CR-0534]

Overview:

CR-0534 identified a problem for an MS/TP Device that supports configuration through Network Port object. The response to CR-0534 is “The Test Conditionality for test 12.1.3.10 will be changed to “This test shall be skipped if the device claims PR-17 or later.” in section 9.1.3.”

Changes:

Checklist Changes

None

Test Plan Changes

[Change Test Conditionality]

9.1 Data Link Layer - MS/TP - Master Node

…

9.1.3 Supports Read Only Max_Master Property

The IUT contains the Max_Master property that is read-only.

135.1-2019 - 12.1.3.10 - Max_Master Test		
	Test Conditionality	~~Must be executed.~~ This test shall be skipped if the IUT claims Protocol_Revision 17 or higher.
	Test Directives	
	Testing Hints	

Test Changes

None

BTL-20.0.1 cr2-12: Fix Destination Virtual Addressing in Local Broadcast Execution Test [BTLWG-1309, CR-0532]

Overview:

14.YY.2.1.2 Change destination virtual address specification (CR-0532)

Changes:

Checklist Changes

None

Test Plan Changes

None

Test Changes

[Reason for Change: Correct Destination Virtual Address per CR-0532]

[BTL Specified Tests]
[modify test as shown]

14.YY.2.1.2 Local Broadcast Execution Test

…

```
3.  RECEIVE PORT (D4-IUT hub WebSocket),
        Encapsulated-NPDU,
        'Originating Virtual Address' =          (IUT's VMAC)
        ~~'Destination Virtual Address' =          (D4's VMAC or X'FFFFFFFF', the local broadcast VMAC)~~
        'Destination Virtual Address' =          (absent or X'FFFFFFFF', the local broadcast VMAC)

        -- 'Destination Options' absent
        'Data Options' =                         ({X'41'}),        -- Secure Path
        'Payload'
            I-Am-Request,
            'I Am Device Identifier' =           (the IUT's Device object),
            'Max APDU Length Accepted' =         (the value specified in the EPICS),
            'Segmentation Supported' =           (the value specified in the EPICS),
            'Vendor Identifier' =                (the identifier registered for this vendor)
```

…

BTL-20.0.1 cr2-13: Update Conditionality for Network Port Configuration Conflict Test [BTLWG-1323, CR-0530]

Overview:

Test conditionality for 7.3.2.X62.1.4 should allow skipping when NPO has no writable properties.
The Test Conditionality should be changed to:
If the IUT does not support any NPO with writable properties, this test shall be skipped.

Changes:

Checklist Changes

None

Test Plan Changes

[Modify section 3.56.1 Base Requirements]

BTL - 7.3.2.X62.1.4 - Network Port Configuration Conflict Test		
	Test Conditionality	~~If the IUT supports WriteProperty, the test must be executed.~~ If the IUT does not support any Network Port objects with writable properties, this test shall be skipped.
	Test Directives	
	Testing Hints	Note that almost all Network Port objects have mandated writable properties, so take care to verify that an IUT which claims no writable properties in its Network Port objects is allowed to make such a claim.

Test Changes

None

BTL-20.0.1 cr2-14: Fix Requirements for AE-LS-B [BTLWG-1342]

Overview:

The 'Supports the CHANGE_OF_LIFE_SAFETY Algorithm in Event_Parameters' option is moved to be combined with the 'Supports the Event Enrollment object supporting the CHANGE_OF_LIFE_SAFETY Algorithm.

Changes:

Checklist Changes

[modify AE-LS-B Checklist section, including removing "Supports the CHANGE_OF_LIFE_SAFETY Algorithm in Event_Parameters"]

Alarm and Event Management - LifeSafety - B		
	R	Base Requirements
	R	Supports the Notification Class Object
	R	Supports AE-INFO-B
	C[1]	Implements intrinsic alarming
	~~R~~	~~Supports the CHANGE_OF_LIFE_SAFETY Algorithm in Event_Parameters~~
	C[1]	Supports the Event Enrollment object using the CHANGE_OF_LIFE_SAFETY Algorithm
	C[2]	Supports AE-ACK-B
	C[3]	Generates event notifications with timestamps of the BACnetDateTime form
	C[3]	Generates event notifications with timestamps of the Time form
	C[3]	Generates event notifications with timestamps of the Sequence Number form
	O	Supports mode transition when Event State is maintained
	O	Supports Event_Message_Texts property
	O	Supports Event_Message_Texts_Config property

Test Plan Changes

[Modify section 5.22.5, merge with 5.22.6, and renumber all following sections in 5.22]

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 Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for CHANGE_OF_RELIABILITY in the EventEnrollment Objects in AE-N-I-B.
	Testing Hints	

~~5.22.5 Supports the CHANGE_OF_LIFE_SAFETY Algorithm in Event_Parameters~~

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

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for the CHANGE_OF_LIFE_SAFETY algorithm in AE-N-I-B
	Testing Hints	

Test Changes

None