BACnet® TESTING LABORATORIES ADDENDA

Addendum fix2 to BTL Test Package 18.1

Revision v4
Revised 7/6/2021

Approved by the BTL Working Group on June 10, 2021;
Approved by the BTL Working Group Voting Members on July 7, 2021;
Published on July 13, 2021.

[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.

BTL-18.1-fix2-1: Fix Who-Has Tests to use BEFORE instead of WAIT [BTLWG-158, CR-0381] 2
BTL-18.1-fix2-2: Allow OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED response [BTLWG-518, CR-0439] 11
BTL-18.1-fix2-3: Response to CR-0410 Test Changes [BTLWG-603, CR-0410] 15
BTL-18.1-fix2-4: Added Definition for Reset [BTLWG-608, CR-0387] 18
BTL-18.1-fix2-5: Add Verify of Setup to Trigger Verification Test [BTLWG-855] 19
BTL-18.1-fix2-6: Lifetime Parameter Value Range [BTLWG-886, CR-0481] 21
BTL-18.1-fix2-7: Removed - already included in Addenda Fix1 22
BTL-18.1-fix2-8: Move WP-B check for Writable Lists to DM-LM-B [BTLWG-947] 23
BTL-18.1-fix2-9: Conditionality of Database Revision Tests [BTLWG-958, CR-0217] 24
BTL-18.1-fix2-10: External Writes Test [BTLWG-967, CR-0262] 25
BTL-18.1-fix2-11: Review and Adjust BTL-R Entries [BTLWG-976] 27
BTL-18.1-fix2-12: DM-RD-B Configuration Requirements for Negative Tests [BTLWG-996] 29
BTL-18.1-fix2-13: Silenced Property Test does not Provide Enough Direction [BTLWG-1007, CR-0476] 30
BTL-18.1-fix2-14: DS-COV-B and DS-COVP-B Test Changes [BTLWG-1021, CR-0484] 31
BTL-18.1-fix2-15: Clarify Data Link Options in Checklist [BTLWG-1043] 35
BTL-18.1-fix2-16: Resizing a Writable Fixed Array Property Test Changes [BTLWG-1069, CR-0488] 40
BTL-18.1-fix2-17: Internal Processing Fail Time [BTLWG-1085] 41
BTL-18.1-fix2-18: Load Control Level Test Fixes [BTLWG-1091, CR-0494] 42

In the following document, language to be added to existing clauses within the BTL Test Package 18.1 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-18.1-fix2-1: Fix Who-Has Tests to use BEFORE instead of WAIT [BTLWG-158, CR-0381]

Overview:
Unconfirmed Response Fail Time was invented at BTL-WG in 2010, and added into tests in sections 9.33.1, 9.33.2 and 13.5. All the tests written prior to that--such as the 9.32.1.12 and other tests of Who-Has execution in section 9.32, did not use this timeout.

Changes:

BTL Checklist Changes

None

BTL Test Plan Changes

None

BTL Specified Test Changes

[In BTL Specified Tests, modify tests]

9.32 Who-Has Service Execution Tests

The purpose of this test group is to verify the correct execution of the Who-Has service request.

Dependencies: None.

BACnet Reference Clause: 16.9.

9.32.1 Execution of Who-Has Service Requests Originating from the Local Network

The purpose of this test group is to verify the correct execution of the Who-Has request service procedure for messages originating from the local network.

9.32.1.1 Object ID Version with No Device Range

Reason for Change: Modified test to remove dependency on EPICS values. The allowance for Unicast I-Have is added. Corrected Timeout used and changed to use BEFORE instead of WAIT.

Purpose: To verify that the IUT can correctly respond to a local broadcast Who-Has service request that utilizes the object identifier form and does not restrict device ranges.

Configuration Requirements: Choose any object (Object1) that exists within the IUT.

Test Steps:

1. *READ V1 = (Object1), Object_Name*
2. TRANSMIT
   DA = LOCAL BROADCAST,
   SA = TD,
   Who-Has-Request,
   'Object Identifier' = *Object1*~~(any object identifier specified in the EPICS)~~
3. ~~WAIT~~*BEFORE* ~~**Internal Processing Fail Time**~~ ***Unconfirmed Response Fail Time***
   RECEIVE
   DA = LOCAL BROADCAST | GLOBAL BROADCAST | *TD*,
   ~~SA = IUT,~~
   I-Have-Request,
   'Device Identifier' = (the IUT's Device object),
   'Object Identifier' = *Object1*~~(the object identifier specified in step 1),~~
   'Object Name' = *V1*~~(the object name specified in the EPICS for this object)~~

9.32.1.2 Object Name Version with no Device Range

Reason for Change: Modified test to remove dependency on EPICS values. The allowance for Unicast I-Have is added. Corrected Timeout used and changed to use BEFORE instead of WAIT.

Purpose: To verify that the IUT can correctly respond to a local broadcast Who-Has service request that utilizes the object name form and does not restrict device ranges.

Configuration Requirements: Choose any object (Object1) that exists within the IUT.

Test Steps:

1. *READ V1 =(Object1), Object_Name*
2. TRANSMIT
   DA = LOCAL BROADCAST,
   SA = TD,
   Who-Has-Request,
   'Object Name' = *V1*~~(any object name specified in the EPICS)~~
3. ~~WAIT~~*BEFORE* ~~**Internal Processing Fail Time**~~ ***Unconfirmed Response Fail Time***
   RECEIVE
   DA = LOCAL BROADCAST | GLOBAL BROADCAST | *TD*,
   ~~SA = IUT,~~
   I-Have-Request,
   'Device Identifier' = (the IUT's Device object),
   'Object Identifier' = *Object1*~~(the object identifier specified in the EPICS for this object)~~,
   'Object Name' = *V1*~~(the object name specified in step 1)~~

9.32.1.3 Object ID Version with IUT Inside of the Device Range

Reason for Change: Modified test to remove dependency on EPICS values. The allowance for Unicast I-Have is added. Corrected Timeout used and changed to use BEFORE instead of WAIT.

Purpose: To verify that the IUT can correctly respond to a local broadcast Who-Has service request that utilizes the object identifier form and specifies a device range restriction that includes the IUT.

Configuration Requirements: Choose any object (Object1) that exists within the IUT.

Test Steps:

1. *READ V1 =(Object1), Object_Name*
2. TRANSMIT
   DA = LOCAL BROADCAST,
   SA = TD,
   Who-Has-Request,
   'Device Instance Low Limit' = (any value L: 0 ≤ L < the Device object instance number of the IUT),
   'Device Instance High Limit' = (any value H~~;~~: H > the Device object instance number of the IUT),
   'Object Identifier' = *Object1*~~(any object identifier specified in the EPICS)~~,
3. ~~WAIT~~*BEFORE* ~~**Internal Processing Fail Time**~~ ***Unconfirmed Response Fail Time***
   RECEIVE
   DA = LOCAL BROADCAST | GLOBAL BROADCAST | *TD*,
   ~~SA = IUT,~~
   I-Have-Request,
   'Device Identifier' = (the IUT's Device object),
   'Object Identifier' = *Object1*~~(the object identifier specified in step 1)~~,
   'Object Name' = *V1*~~(the object name specified in the EPICS for this object)~~

9.32.1.4 Object ID Version with IUT Outside of the Device Range

Reason for Change: Modified test to remove dependency on EPICS values. Corrected Timeout used and changed to use BEFORE instead of WAIT.

Purpose: To verify that the IUT ignores a local broadcast Who-Has service request that utilizes the object identifier form and specifies a device range restriction that does not include the IUT.

Configuration Requirements: Choose any object (Object1) that exists within the IUT.

Test Steps:

1. TRANSMIT
   DA = LOCAL BROADCAST,
   SA = TD,
   Who-Has-Request,
   'Device Instance Low Limit' = (any value > 0: the Device object instance number does not fall in the range between Device Instance Low Limit and Device Instance High Limit),
   'Device Instance High Limit' = (any value > Device Instance Low Limit: the Device object instance number does not fall in the range between Device Instance Low Limit and Device Instance High Limit),
   'Object Identifier' = *Object1*~~(any object identifier specified in the EPICS)~~
2. WAIT ~~**Internal Processing Fail Time**~~ ***Unconfirmed Response Fail Time***
3. CHECK (verify that the IUT does not respond)

9.32.1.5 Object Name Version with IUT Inside of the Device Range

Reason for Change: Modified test to remove dependency on EPICS values. The allowance for Unicast I-Have is added. Corrected Timeout used and changed to use BEFORE instead of WAIT.

Purpose: To verify that the IUT can correctly respond to a local broadcast Who-Has service request that utilizes the object name form and specifies a device range restriction that includes the IUT.

Configuration Requirements: Choose any object (Object1) that exists within the IUT.

Test Steps:

1. *READ V1 =(Object1), Object_Name*
2. TRANSMIT
   DA = LOCAL BROADCAST,
   SA = TD,
   Who-Has-Request,
   'Device Instance Low Limit' = (any value L: 0 ≤ L < the Device object instance number of the IUT),
   'Device Instance High Limit' = (any value H: H > the Device object instance number of the IUT),
   'Object Name = *V1*~~(any object name specified in the EPICS)~~
3. ~~WAIT~~*BEFORE* ~~**Internal Processing Fail Time**~~ ***Unconfirmed Response Fail Time***
   RECEIVE
   DA = LOCAL BROADCAST | GLOBAL BROADCAST | *TD*,
   ~~SA = IUT,~~
   I-Have-Request,
   'Device Identifier' = (the IUT's Device object),
   'Object Identifier' = *Object1*~~(the object identifier specified in the EPICS for this object)~~,
   'Object Name' = *V1*~~(the object name specified in step 1)~~

9.32.1.7 Object ID Version with IUT Device Instance Equal to the High Limit of the Device Range

Reason for Change: Modified test to remove dependency on EPICS values. The allowance for Unicast I-Have is added. Corrected Timeout used and changed to use BEFORE instead of WAIT.

Purpose: To verify that the IUT correctly recognizes the high limit of the specified device range for Who-Has service requests that utilize the object identifier form.

Configuration Requirements: Choose any object (Object1) that exists within the IUT.

Test Steps:

1. *READ V1 =(Object1), Object_Name*
2. TRANSMIT
   DA = LOCAL BROADCAST,
   SA = TD,
   Who-Has-Request,
   'Device Instance Low Limit' = (any value L: 0 ≤ L < the Device object instance number of the IUT),
   'Device Instance High Limit' = (The Device object instance number of the IUT),
   'Object Identifier' = *Object1*~~(any object identifier specified in the EPICS)~~
3. ~~WAIT~~*BEFORE* ~~**Internal Processing Fail Time**~~ ***Unconfirmed Response Fail Time***
   RECEIVE
   DA = LOCAL BROADCAST | GLOBAL BROADCAST | *TD*,
   ~~SA = IUT,~~
   I-Have-Request,
   'Device Identifier' = (the IUT's Device object),
   'Object Identifier' = *Object1*~~(the object identifier specified in step 1)~~,
   'Object Name' = *V1*~~(the object name specified in the EPICS for this object)~~

9.32.1.8 Object ID Version with IUT Device Instance Equal to the Low Limit of the Device Range

Reason for Change: Modified test to remove dependency on EPICS values. The allowance for Unicast I-Have is added. Corrected Timeout used and changed to use BEFORE instead of WAIT.

Purpose: To verify that the IUT correctly recognizes the low limit of the specified device range for Who-Has service requests that utilize the object identifier form.

Configuration Requirements: Choose any object (Object1) that exists within the IUT.

Test Steps:

1. *READ V1 =(Object1), Object_Name*
2. TRANSMIT
   DA = LOCAL BROADCAST,
   SA = TD,
   Who-Has-Request,
   'Device Instance Low Limit' = (The Device object instance number of the IUT),
   'Device Instance High Limit' = (any value H: H > the Device object instance number of the IUT),
   'Object Identifier' = *Object1*~~(any object identifier specified in the EPICS)~~
3. ~~WAIT~~*BEFORE* ~~**Internal Processing Fail Time**~~ ***Unconfirmed Response Fail Time***
   RECEIVE
   DA = LOCAL BROADCAST | GLOBAL BROADCAST | *TD*,
   ~~SA = IUT,~~
   I-Have-Request,
   'Device Identifier' = (the IUT's Device object),
   'Object Identifier' = *Object1*~~(the object identifier specified in step 1)~~,
   'Object Name' = *V1*~~(the object name specified in the EPICS for this object)~~

9.32.1.9 Object Name Version with IUT Device Instance Equal to the High Limit of the Device Range

Reason for Change: Modified test to remove dependency on EPICS values. The allowance for Unicast I-Have is added. Corrected Timeout used and changed to use BEFORE instead of WAIT.

Purpose: To verify that the IUT correctly recognizes the high limit of the specified device range for Who-Has service requests that utilize the object name form.

Configuration Requirements: Choose any object (Object1) that exists within the IUT.

Test Steps:

1. *READ V1 =(Object1), Object_Name*
2. TRANSMIT

Who-Has-Request,
'Device Instance Low Limit' = (any value L: 0 ≤ L < the Device object instance number of the IUT),
'Device Instance High Limit' = (The Device object instance number of the IUT),
'Object Name = *V1*~~(any object name specified in the EPICS)~~

3. ~~WAIT~~*BEFORE* ~~**Internal Processing Fail Time**~~ ***Unconfirmed Response Fail Time***
RECEIVE
DA = LOCAL BROADCAST | GLOBAL BROADCAST | *TD*,
~~SA = IUT,~~
I-Have-Request,
'Device Identifier' = (the IUT's Device object),
'Object Identifier' = *Object1*~~(the object identifier specified in the EPICS for this object)~~,
'Object Name' = *V1*~~(the object name specified in step 1)~~

9.32.1.10 Object Name Version with IUT Device Instance Equal to the Low Limit of the Device Range

Reason for Change: Modified test to remove dependency on EPICS values. The allowance for Unicast I-Have is added. Corrected Timeout used and changed to use BEFORE instead of WAIT.

Purpose: To verify that the IUT correctly recognizes the low limit of the specified device range for Who-Has service requests that utilize the object name form.

Configuration Requirements: Choose any object (Object1) that exists within the IUT.

Test Steps:

1. *READ V1 =(Object1), Object_Name*
2. TRANSMIT
DA = LOCAL BROADCAST,
SA = TD,
Who-Has-Request,
'Device Instance Low Limit' = (The Device object instance number of the IUT),
'Device Instance High Limit' = (any value H: H > the Device object instance number of the IUT),
'Object Name = *V1*~~(any object name specified in the EPICS)~~
3. ~~WAIT~~*BEFORE* ~~**Internal Processing Fail Time**~~ ***Unconfirmed Response Fail Time***
RECEIVE
DA = LOCAL BROADCAST | GLOBAL BROADCAST | *TD*,
~~SA = IUT,~~
I-Have-Request,
'Device Identifier' = (the IUT's Device object),
'Object Identifier' = *Object1*~~(the object identifier specified in step 1)~~,

9.32.1.11 Object Name Version, Directed to a Specific MAC Address

Reason for Change: Modified test to remove dependency on EPICS values. The allowance for Unicast I-Have is added. Corrected Timeout used and changed to use BEFORE instead of WAIT.

Purpose: To verify that the IUT responds with a broadcast I-Have service request even if the Who-Has service requests was not transmitted with a broadcast address.

Configuration Requirements: Choose any object (Object1) that exists within the IUT.

Test Steps:

1. *READ V1 =(Object1), Object_Name*
2. TRANSMIT Who-Has-Request,
'Object Name' = *V1*~~(any object name specified in the EPICS)~~,
3. ~~WAIT~~*BEFORE* ~~**Internal Processing Fail Time**~~ ***Unconfirmed Response Fail Time***
RECEIVE
DA = LOCAL BROADCAST | GLOBAL BROADCAST | *TD*,
~~SA = IUT,~~
I-Have-Request,
'Device Identifier' = (the IUT's Device object),

'Object Identifier' = *Object1*~~(the object identifier specified in the EPICS for this object)~~,
'Object Name' = *V1*~~(the object name specified in step 1)~~

9.32.1.12 Who-Has After Object_Name Changed

Reason for Change: The BACnet standard (per addendum 135-2012ar-5) now allows the IUT to send a unicast response. Corrected Timeout used and changed to use BEFORE instead of WAIT.

Dependencies: Who-Has Service Execution Tests, 9.32.1.2
BACnet Reference Clause: 16.9

Purpose: To verify that a device correctly responds to Who-Has service requests after the Object_Name property of an object in the device is changed.

Test Concept: The Object_Name property of the referenced object is read to determine its initial value. The Object_Name property is then changed to a different value, V2, which is not already used by an object in the IUT. The test then verifies correct responses to Who-Has requests that include an 'Object Name' parameter, using the values V1 and V2.

Configuration: An object, O1, exists within the IUT that has a modifiable Object_Name property and has the value V1. If IUT does not support objects with modifiable Object_Name properties, then this test shall be skipped.

Test Steps:

1. READ V1 = O1, Object_Name
2. IF (Object_Name is writable) THEN
   WRITE O1, Object_Name = V2
   ELSE
   MAKE (O1, Object_Name = V2)
3. TRANSMIT
   DESTINATION = GLOBAL BROADCAST,
   Who-Has-Request,
   'Object Name' = V1
4. WAIT ~~**Internal Processing Fail Time**~~ ***Unconfirmed Response Fail Time***
5. CHECK (Verify that the IUT does not respond with an I-Have request)
6. TRANSMIT
   DESTINATION = GLOBAL BROADCAST,
   Who-Has-Request,
   'Object Name' = V2
7. *BEFORE* ***Unconfirmed Response Fail Time***
   RECEIVE ~~DESTINATION~~*DA* = LOCAL BROADCAST | GLOBAL BROADCAST | *TD,*
   I-Have-Request,
   'Device Identifier' = (the IUT's Device object),
   'Object Identifier' = O1,
   'Object Name' = V2

9.32.1.13 Who-Has After Object_Identifier Changed

Reason for Change: The BACnet standard (per addendum 135-2012ar-5) now allows the IUT to send a unicast response. Corrected Timeout used and changed to use BEFORE instead of WAIT.

Dependencies: Who-Has Service Execution Tests, 9.32.1.1

BACnet Reference Clause: 16.9

Purpose: To verify that a device correctly responds to Who-Has service requests after the Object_Identifier property of an object in the device is changed.

Test Concept: The Object_ Identifier property of the referenced object, O1, is verified to contain the value O1. The Object_Identifier property is then changed to a different value, O2, which is not already in use by a different object in the IUT. The test then verifies correct responses to Who-Has requests that include an 'Object Identifier' parameter, using the values O1 and O2.

Configuration: An object, O1, exists within the IUT that has a modifiable Object_ Identifier property. If the IUT does not support objects with modifiable Object_Identifiers, then this test shall be skipped.

Test Steps:
1. VERIFY O1, Object_Identifier = O1
2. IF (O1 is writable) THEN
   WRITE O1, Object_Identifier = O2
   ELSE
   MAKE (O1, Object_Identifier = O2)
3. TRANSMIT
   DESTINATION = GLOBAL BROADCAST,
   Who-Has-Request,
   'Object Identifier' = O1
4. WAIT ~~**Internal Processing Fail Time**~~ ***Unconfirmed Response Fail Time***
5. CHECK (Verify that the IUT does not respond with an I-Have request)
6. TRANSMIT
   DESTINATION = GLOBAL BROADCAST,
   Who-Has-Request,
   'Object Identifier' = O2
7. *BEFORE* ***Unconfirmed Response Fail Time***
   RECEIVE ~~DESTINATION~~*DA* = LOCAL BROADCAST | GLOBAL BROADCAST | *TD,*
   I-Have-Request,
   'Device Identifier' = (the IUT's Device object),
   'Object Identifier' = O2
   'Object Name' = *V1*~~(the object name specified in the EPICS for this object)~~

9.32.2 Execution of Who-Has Service Requests Originating from a Remote Network

9.32.2.1 Object ID Version, Global Broadcast from a Remote Network

Reason for Change: Modified test to remove dependency on EPICS values. The allowance for Unicast I-Have is added. Corrected Timeout used and changed to use BEFORE instead of WAIT.

Purpose: To verify the ability of the IUT to recognize the origin of a globally broadcast Who-Has service request and to respond such that the device originating the request receives the response.

Configuration Requirements: Choose any object (Object1) that exists within the IUT.

Test Steps:
1. *READ V1 =(Object1), Object_Name*
2. TRANSMIT
~~DESTINATION = LOCAL BROADCAST,~~
~~SA = TD,~~
   DNET = GLOBAL BROADCAST,
   SNET = (*X:* any remote network number),
   SADR = (*Y:* any MAC address valid for the specified network),
   Who-Has-Request,
   'Object Identifier' = *Object1*~~(any object identifier specified in the EPICS)~~
3. ~~WAIT~~*BEFORE* ~~**Internal Processing Fail Time**~~ ***Unconfirmed Response Fail Time***
   RECEIVE
   DESTINATION = GLOBAL BROADCAST | REMOTE BROADCAST (to the network *X*~~specified in step 1~~) | *TD (DNET = X, DADR = Y)*,
   I-Have-Request,
   'Device Identifier' = (the IUT's Device object),
   'Object Identifier' = *Object1*~~(the object identifier specified in step 1)~~,
   'Object Name' = *V1*~~(the object name specified in the EPICS for this object)~~

9.32.2.2 Object ID Version, Remote Broadcast

Reason for Change: Modified test to remove dependency on EPICS values. The allowance for Unicast I-Have is added. Corrected Timeout used and changed to use BEFORE instead of WAIT.

Purpose: To verify the ability of the IUT to recognize the origin of a remotely broadcast Who-Has service request and to respond such that the device originating the request receives the response.

Configuration Requirements: Choose any object (Object1) that exists within the IUT.

Test Steps:
1. *READ V1 =(Object1), Object_Name*
2. TRANSMIT
~~DESTINATION = LOCAL BROADCAST,~~
~~SA = TD,~~
SNET = (any remote network number),
SADR = (any MAC address valid for the specified network),
Who-Has-Request,
'Object Identifier' = *Object1*~~(any object identifier specified in the EPICS)~~
3. ~~WAIT~~*BEFORE* ~~**Internal Processing Fail Time**~~ ***Unconfirmed Response Fail Time***
RECEIVE
DESTINATION = GLOBAL BROADCAST | REMOTE BROADCAST (to the network *X*~~specified in step 1~~) | *TD (DNET = X, DADR = Y)*,
I-Have-Request,
'Device Identifier' = (the IUT's Device object),
'Object Identifier' = *Object1*~~(the object identifier specified in step 1)~~,
'Object Name' = *V1*~~(the object name specified in the EPICS for this object)~~

9.32.2.X3 Who-Has for Non-existent Object_Name

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

Purpose: Verifies correct responses to Who-Has service requests by 'Object Name' when the object does not exist in the IUT.

Test Concept: The test verifies the correct non-response to Who-Has service request with 'Object Name' when that named object does not exist in the IUT.

Configuration Requirements: Choose any character string value V1, which is not the Object_Name of any object in the IUT. The IUT shall be placed in a state where it is not producing I-Have spontaneously.

Test Steps:

1. TRANSMIT Who-Has-Request,
'Object Name' = V1
2. WAIT **Unconfirmed Response Fail Time**
3. CHECK (the IUT does not respond with an I-Have request with 'Object Name' containing V1)

9.32.2.X5 Who-Has for Non-existent Object_Identifier

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

Purpose: Verifies correct responses to Who-Has service requests when the object does not exist in the IUT.

Test Concept: The test verifies the correct non-response to Who-Has request with that 'Object Identifier' parameter for an object which does not exist.

Configuration Requirements: Choose any standard object (Object1) that does not exist within the IUT, i.e. any unsupported Object Type or any supported Object Type for which the instance does not exist. The IUT shall be placed in a state where it is not producing I-Have spontaneously.

Test Steps:

1. TRANSMIT ReadProperty-Request,
'Object Identifier' = Object1,
'Property Identifier' = Object_Identifier
2. RECEIVE BACnet-Error-PDU,

'Error Class' = OBJECT,
'Error Code' = UNKNOWN_OBJECT

3. TRANSMIT Who-Has-Request,
'Object Identifier' = Object1
4. WAIT **Unconfirmed Response Fail Time**
5. CHECK (the IUT does not respond with an I-Have request with 'Object Identifier' containing Object1)

BTL-18.1-fix2-2: Allow OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED response [BTLWG-518, CR-0439]

Overview:
BTL-CR-0439 ruled for Event_Parameters and Fault_Parameters, the Error Class: PROPERTY Error Code: OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED is a valid response. The clarification address specifically mentioned tests: 9.22.2.4, 9.23.2.7 and 8.5.17 the CHANGE_OF_RELIABILITY Tests.

Changes:

BTL Checklist Changes

None

BTL Test Plan Changes

[In BTL Test Plan, add new test to base requirements of DS-WP-B]

4.6.1 Base Requirements

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

…		
BTL - 9.22.2.X4 - Writing with a Property Value related to a not supported optional functionality		
	Test Conditionality	If the IUT does not support a writable property with an explicit mention BACnet-Error-code OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED, then this test shall be skipped.
	Test Directives	For each writable property that is related to an optional functionality that the object is not supported
	Testing Hints	135-2020 clause 12. (12.12.8,12.24.10,12.25.26,12.30.11,12.53.11,12.56.14,12.57.21,12.62.14)

[In BTL Test Plan, add new test to base requirements of DS-WPM-B]

4.8.1 Base Requirements

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

…		
BTL - 9.23.2.X7 - Writing with a Property Value related to a not supported optional functionality		
	Test Conditionality	If the IUT does not support a writable property with an explicit mention BACnet-Error-code OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED, then this test shall be skipped.
	Test Directives	For each writable property that is related to an optional functionality that the object is not supported
	Testing Hints	135-2020 clause 12. (12.12.8,12.24.10,12.25.26,12.30.11,12.53.11,12.56.14,12.57.21,12.62.14)

BTL Specified Test Changes

[In BTL Specified Tests, modify test 9.22.2.4]

9.22.2.4 Writing with a Property Value that is Out of Range

Reason for Change: Modified test to remove dependency on EPICS values. *Modified to allow OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED in some specific case.*

Purpose: To verify that the IUT can execute WriteProperty service requests when an attempt is made to write a value that is outside of the supported range.

Test Concept: The TD attempts to write to a property using a value that is outside of the supported range. If the IUT does not contain any writable properties that have restricted ranges, then this test shall be skipped.

Test Steps:

1. READ X = (Object1), P1
2. TRANSMIT WriteProperty-Request,
   'Object Identifier' = (Object1, any object with writable properties),
   'Property Identifier' = (P1, any writable property with a restricted range of values),
   'Property Value' =(any value, of the correct datatype, that is outside the supported range)
3. IF (*Protocol_Revision is present* and Protocol_Revision >= 4) THEN
   RECEIVE
   *(*BACnet-Error-PDU,
   'Error Class' = PROPERTY,
   'Error Code' = VALUE_OUT_OF_RANGE) |
   (BACnet-Error-PDU,
   'Error Class' = PROPERTY,
   'Error Code' = OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED)
   ELSE
   RECEIVE (BACnet-Error-PDU,
   'Error Class' = PROPERTY,
   'Error Code' = VALUE_OUT_OF_RANGE) |
   (BACnet-Error-PDU,
   'Error Class' = PROPERTY,
   'Error Code' = OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED) |
   (BACnet-Reject-PDU,
   Reject Reason = PARAMETER_OUT_OF_RANGE)
4. VERIFY (Object1), P1 = X

Notes to-*Tester*: The value used in step 2 shall be of the correct datatype. For bit string types, the bit count shall be correct, for Date and Time values, the value shall be within the range defined by the standard for the datatype, for constructed values, the constructed value shall match the structure defined by the ASN.1 and all field values shall be within the ranges defined by the standard for those field values.
In this test, the error code OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED is permitted only if the value written would require the device to exhibit non-supported optional functionality.

[In BTL Specified Tests, Add new test 9.22.2.X4]

9.22.2.X4 Writing a Property Value Related to Non-supported Optional Functionality

Reason for Change: No test exists in the actual test plan when a strong indication by the explicit mention of PROPERTY OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED is mentioned in Clause 12.

Purpose: To verify that the IUT responds correctly to WriteProperty service requests when the value provided would require the device to exhibit non-supported optional functionality.

Test Concept: A writable property, P1, is selected for which the standard defines optional behavior which the IUT does not support, and the writing of specific values to the property would require the device to exhibit the non-supported behavior. The TD attempts to write to the property using a selected value, V1, which would require the IUT to exhibit non-supported optional functionality.

Test Steps:

1. READ X = (Object1), P1
2. TRANSMIT WriteProperty-Request,
   'Object Identifier' = (Object1, the object containing a writable P1),

'Property Identifier' = (P1, any writable property with a restricted range of values),
'Property Value' = (V1, a value that would require a non-supported optional functionality)

3. RECEIVE BACnet-Error-PDU
'Error Class' = PROPERTY,
'Error Code' = OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED
4. VERIFY (Object1), P1 = X

[In BTL Specified Tests, Modify test 9.23.2.7]

9.23.2.7 Writing with a Property Value that is Out of Range

Reason for Change: Modified test to remove dependency on EPICS values. Modified to allow this test to be used on all protocol revisions. *Modified to allow OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED in some specific case.*

Purpose: This test case verifies that the IUT can execute WritePropertyMultiple service requests when an attempt is made to write a value that is outside of the supported range~~. This test shall only be performed if Protocol_Revision is present and has a value greater than or equal to 4.~~

Test Concept: The TD shall select an object, designated Object1, in the IUT that contains a writable property designated P1. The TD attempts to write to a property using a value that is outside of the supported range.

Note to Tester: In this test, the error code OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED is permitted only if the value written would require the device to exhibit non-supported optional functionality.

Test Steps:

1. READ X = (Object1), P1
~~1. VERIFY (Object1), P1 = (the value defined for this property in the EPICS),~~
2. TRANSMIT WritePropertyMultiple-Request,
'Object Identifier' = (Object1, any object with writable properties),
'Property Identifier' = (P1, any property with a restricted range of values),
'Property Value' = (any value that is outside the supported range)
3. *IF (Protocol_Revision < 4) THEN*
RECEIVE
(WritePropertyMultiple-Error,
'Error Class' = PROPERTY,
'Error Code' = VALUE_OUT_OF_RANGE,
'Object Identifier' = Object1,
'Property Identifier' = P1) |
(WritePropertyMultiple-Error,
'Error Class' = PROPERTY,
'Error Code' = OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED
'Object Identifier' = Object1,
'Property Identifier' = P1) |
(BACnet-Reject-PDU,
'Reject Reason' = PARAMETER_OUT_OF_RANGE)
ELSE
RECEIVE
(WritePropertyMultiple-Error,
'Error Class' = PROPERTY,
'Error Code' = VALUE_OUT_OF_RANGE,
'Object Identifier' = Object1,
'Property Identifier' = P1) |
(WritePropertyMultiple-Error,
'Error Class' = PROPERTY,
'Error Code' = OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED
'Object Identifier' = Object1,
'Property Identifier' = P1)
4. VERIFY (OBJECT1), P1 = *X~~(the value defined for this property in the EPICS)~~*

[In BTL Specified Tests, add new test 9.23.2.X7]

9.23.2.X7 Writing with a Property Value that is Out of the restricted range

Reason for Change: No test exists in the actual test plan when a strong indication by the explicit mention of PROPERTY OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED is mentioned in Clause 12.

Purpose: To verify that the IUT can execute WritePropertyMultiple service requests when an attempt is made to write a value that would require a non-supported optional functionality

Test Concept: The TD attempts to write to a property using a value that would require a non-supported optional functionality.

Test Steps:

1. READ X = (Object1), P1
2. TRANSMIT WritePropertyMultiple-Request,
   'Object Identifier' = (Object1, any object with writable properties),
   'Property Identifier' = (P1, any property with a restricted range of values),
   'Property Value' = (a value that would require a non-supported optional functionality)
3. RECEIVE WritePropertyMultiple-Error,
   'Error Class' = PROPERTY,
   'Error Code' = OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED
   'Object Identifier' = Object1,
   'Property Identifier' = P1
4. VERIFY (Object1), P1 = X

BTL-18.1-fix2-3: Response to CR-0410 Test Changes [BTLWG-603, CR-0410]

Overview:
CR-0410 response specified that in each of tests 9.20.1.7, 9.20.1.8, and 9.20.19 to add another Note to Tester checking that no unexpected properties appear in the returned result

Changes:

BTL Checklist Changes

None

BTL Test Plan Changes

[In BTL Test Plan, modify section 4.4.1 reference to test 9.20.1.7 from 135.1-2019 to BTL]

4.4.1 Base Requirements

…

***BTL~~135.1-2019~~* - 9.20.1.7 - Reading ALL Properties**		
	Test Conditionality	Must be executed. This test shall be skipped for any object type whose set of properties cannot be transmitted in the largest supported response message based on the IUT's APDU and segmentation limitations.
	Test Directives	
	Testing Hints	The pre-tester should apply this test to every object type. If the set of properties differs between instances of the same object type in the IUT, each form of the object type should be tested.

…

BTL Specified Test Changes

[In BTL Specified Tests, add test 9.20.1.7 from 135.1-2019 and modify]

9.20.1.7 Reading ALL Properties

Reason for Change: Modified test to remove dependency on EPICS values. Addendum 135-2008x. Addendum 135-2010ao-5.

Purpose: To verify the ability to correctly execute a ReadPropertyMultiple service request that uses the special property identifier ALL. One instance of each object-type supported is tested.

Test Steps:

1. REPEAT ObjectX = (one instance of each supported object type) DO {
    TRANSMIT ReadPropertyMultiple-Request,
        'Object Identifier' = ObjectX,
        'Property Identifier' = ALL
    RECEIVE ReadPropertyMultiple-ACK,
        'Object Identifier' = ObjectX,
        'List Of Results' = (a list of all standard properties documented for ObjectX in the EPICS plus all proprietary properties present in ObjectX, each with a valid value, excluding the Property_List property)
    ~~REPEAT P = (each property supported by Object1) DO {~~
        ~~'Property Identifier' = P,~~
        ~~'Property Value' = (the value of P specified in the EPICS)~~
    ~~}~~
}

Notes to Tester: ~~Any proprietary properties that are supported for the object-type shall also be returned (see BACnet 15.7.3.1.2).~~ If a property which is not readable using the ReadPropertyMultiple service is in the specified object, and Protocol_Revision < 7, then either no entry is returned, or an error code is returned. If Protocol_Revision >= 7, then the entry shall contain 'Error Class': PROPERTY and 'Error-Code': READ_ACCESS_DENIED for that property. Property_List (371) shall not appear in the List of Results.

[In BTL Specified Tests, modify tests 9.20.1.8]

9.20.1.8 Reading OPTIONAL Properties

Reason for Change: Modified test to remove dependency on EPICS values. Addendum 135-2008x. Added language to clarify the properties required to be returned.

Purpose: To verify the ability to correctly execute a ReadPropertyMultiple service request that uses the special property identifier OPTIONAL *by returning all of the object's optional properties.* ~~One instance of each object-type supported is tested.~~

Test Steps:

1. REPEAT ObjectX = (one instance of each supported object type) DO {
    TRANSMIT ReadPropertyMultiple-Request,
        'Object Identifier' = ~~Object1~~ObjectX,
        'Property Identifier' = OPTIONAL
    RECEIVE ReadPropertyMultiple-ACK,
        'Object Identifier' = ~~Object1~~ObjectX,
        'List Of Results' = (a list of all standard properties with a conformance code of O documented for ObjectX in the EPICS, each with a valid value)
    ~~REPEAT P = (each optional property supported by Object1) DO {~~
        ~~'Property Identifier' = P,~~
        ~~'Property Value' = (the value of P specified in the EPICS)~~
    ~~}~~
}

Notes to Tester: If no optional properties are supported then an empty 'List of Results' shall be returned for the specified property. *If a property which is not readable using the ReadPropertyMultiple service is in the specified object, and Protocol_Revision < 7, then either no entry is returned, or an error code is returned. If Protocol_Revision >= 7, then the entry shall contain Error Class: PROPERTY and 'Error-Code': READ_ACCESS_DENIED for that property.*

[In BTL Specified Tests, Add test 9.20.1.9 from 135.1-2021 and modify test]

9.20.1.9 Reading REQUIRED Properties

Reason for Change: Modified test to remove dependency on EPICS values. Addendum 135-2008x. Addendum 135-2010ao-5. Added language to clarify the properties required to be returned.

Purpose: To verify the ability to correctly execute a ReadPropertyMultiple service request that uses the special property identifier REQUIRED *by returning all of the object's required properties.* ~~One instance of each object-type supported is tested. The property identifier REQUIRED means that only those standard properties having a conformance code of "R" or "W" shall be returned.~~

Test Steps:

1. REPEAT ObjectX = (one instance of each supported object type) DO {
TRANSMIT ReadPropertyMultiple-Request,
'Object Identifier' = ObjectX,
'Property Identifier' = REQUIRED
RECEIVE ReadPropertyMultiple-ACK,
'Object Identifier' = ObjectX,
'List Of Results' = (a list of all standard properties with a conformance code of R or W documented for ObjectX in the EPICS, each with a valid value, excluding the Property_List property)
~~REPEAT P = (each property supported by Object1) DO {~~
~~'Property Identifier' = P,~~
~~'Property Value' = (the value of P specified in the EPICS)~~
~~}~~
}

Notes to Tester: If a property which is not readable using the ReadPropertyMultiple service is in the specified object, and Protocol_Revision < 7, then either no entry is returned, or an error code is returned. If Protocol_Revision >= 7, then the entry shall contain 'Error Class': PROPERTY and ‘Error-Code’: READ_ACCESS_DENIED for that property. Property_List (371) shall not appear in the List of Results.

BTL-18.1-fix2-4: Added Definition for Reset [BTLWG-608, CR-0387]

Overview:
There are tests like 14.3.3 with steps that require to “MAKE (the IUT reset)”. But there is no specification what kind of reset is to be applied to the IUT. As the term “reset” is not yet defined in the test standard, it shall be added to the definitions section of the BTL Specified Tests.

Changes:

BTL Checklist Changes

None

BTL Test Plan Changes

None

BTL Specified Test Changes

[In BTL Specified Tests, append to 3. DEFINITIONS]

3.x Common language used in tests

Reason for Change: New section introduced

‘any valid value’ - Any valid value refers to any value of the correct data type and within the vendor’s range specified for the property this is applied to.
‘any appropriate password’ - Any password that meets the Configuration Requirements specified in the test or test section. Passwords when required by the vendor are required to be no more than 20 characters.
‘reset’ - Some tests require to reset the IUT. Reset includes power cycle via switch, power cycle via loss of power and reinitializeDevice WARMSTART. As defined by the BACnet standard, “WARMSTART shall mean to reboot the device and start over, retaining all data and programs that would normally be retained during a brief power outage.”

BTL-18.1-fix2-5: Add Verify of Setup to Trigger Verification Test [BTLWG-855]

Overview:
Added test steps to verify Log_Interval property shall be zero when Logging_Type property has value TRIGGERED.

Changes:

BTL Checklist Changes

None

BTL Test Plan Changes

None

BTL Specified Test Changes

[In BTL Specified Tests, modify the test 7.3.2.24.19]

7.3.2.24.19 Trigger Verification Test

Reason for Change: Modified test to check Log_Interval shall be zero when Logging_Type property has either of the values COV or TRIGGERED. Updated Configuration Requirements.

~~Dependencies: ReadRange Service Execution, 9.21;~~

~~BACnet Reference Clause: 12.25.27, 12.30.12~~

Purpose: To verify logged samples are based on the triggered Logging_Type.

Test Concept: The log, *O1* is configured to log based on TRIGGERED. Logging is enabled. After a period, the buffer is checked to verify the data in the buffer is based on triggered values.

Configuration Requirements: ~~The IUT shall be configured such that the monitored object's Logging_Type is set to TRIGGERED.~~ *The object being tested shall be configured with Logging_Type set to TRIGGERED.*

Test Steps:

1. VERIFY Logging_Type = TRIGGERED
2. VERIFY Log_Interval = 0
3~~1~~. WRITE Enable = FALSE
4~~2~~. WRITE Record_Count = 0 ~~results in a buffer purged record~~
5~~3~~. WRITE Enable = TRUE ~~results in a logging enable record~~
6~~4~~. WAIT (10 seconds)
7~~5~~. WRITE Trigger = TRUE
8~~6~~. WAIT (20 seconds)
9~~7~~. WRITE Trigger = TRUE
10~~8~~. WAIT (40 seconds)
11~~9~~. WRITE Trigger = TRUE
12~~10~~. WAIT (30 seconds)
13~~11~~. WRITE Enable = FALSE ~~-- results in a logging disabled record~~
~~12. VERIFY RecordCount = 6~~
14. READ N = Record_Count
15~~13~~. REPEAT X = (1 through *34*)
TRANSMIT ReadRange-*Request*

'Object Identifier' = O_1,
'Property Identifier' = Log_Buffer,
'Reference Index' = *N-4*+X,
'Count' = 1

RECEIVE ~~ReadRangeAck~~*ReadRange-ACK*
'Object Identifier' = O_1,
'Property Identifier' = Log_Buffer,
'Result Flags' = (~~?~~*False, ~~?~~False*, False),
'Item Count' = 1,
'Item Data' = ~~(~~(one data record storing the timestamp in TS[X])~~)~~

16. TRANSMIT ReadRange-Request
'Object Identifier' = O_1,
'Property Identifier' = Log_Buffer,
'Reference Index' = N,
'Count' = 1

*RECEIVE ReadRange-*ACK~~ck~~
'Object Identifier' = O1,
'Property Identifier' = Log_Buffer,
'Result Flags' = (False, True, False),
'Item Count' = 1,
'Item Data' = (one data record storing the timestamp in TS[4])

~~14. CHECK(TS[3] - TS[2] ~= 10 seconds)~~
~~15. CHECK(TS[4] - TS[3] ~= 20 seconds)~~
~~16. CHECK(TS[5] - TS[4] ~= 40 seconds)~~
~~17. CHECK(TS[6] - TS[5] ~= 30 seconds)~~
17. CHECK(TS[2] - TS[1] ~= 20 seconds)
18. CHECK(TS[3] - TS[2] ~= 40 seconds)
19. CHECK(TS[4] - TS[3] ~= 30 seconds)

BTL-18.1-fix2-6: Lifetime Parameter Value Range [BTLWG-886, CR-0481]

Overview:
As noted in CRR-0481, the test requires selecting a value larger than the supported lifetime but with the range of supported unsigned values. If the IUT allows a lifetime of the full range, then this test cannot be executed.

Changes:

BTL Checklist Changes

None

BTL Test Plan Changes

[Modify section 4.10.1]

BTL - 9.10.2.X3 - The LifeTime Parameter is Out of Range		
	Test Conditionality	~~Must be executed.~~ *If the lifetime parameter accepts values across the full range of unsigned values decodable by the IUT (which must be at least the full range of Unsigned16), then this test shall be skipped.*
	Test Directives	
	Testing Hints	

BTL Specified Test Changes

None

BTL-18.1-fix2-7: Removed - already included in Addenda Fix1

BTL-18.1-fix2-8: Move WP-B check for Writable Lists to DM-LM-B [BTLWG-947]

Overview:
Currently, the checklist Supports DS-WP-B is a separate mandatory requirement of claiming Device Management - List Manipulation - B BIBB. Base Requirements section would be better place for DS-WP-B in Device Management - List Manipulation – B BIBB of checklist.

Changes:

BTL Checklist Changes

Device Management - List Manipulation - B		
	R	Base Requirements
	R	All writable list properties in the IUT support list manipulation
	~~R[1]~~	~~Supports DS-WP-B~~
~~[1] A writable list property is one which is modifiable via any BACnet service.~~		

BTL Test Plan Changes

8.24.1 Base Requirements

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

A device that supports DM-LM-B is required to support AddListElement and RemoveListElement on every writable list property. In addition, any list property modifiable via AddListElement shall be modifiable via WriteProperty.

…		
Verify Checklist		
	Test Conditionality	*Must be executed.*
	Test Directives	*Verify that the IUT claims support for DS-WP-B and that the DS-WP-B option "Contains Writable List Properties" is claimed.*
	Testing Hints	

~~8.24.3 Supports DS-WP-B~~

~~Any device that contains a writable list property and that supports the list manipulation routines must also support the Write Property service.~~

~~**Verify Checklist**~~		
	~~**Test Conditionality**~~	~~Must be executed.~~
	~~**Test Directives**~~	~~Verify that the IUT claims support for DS-WP-B.~~
	~~**Testing Hints**~~	

BTL Specified Test Changes

None

BTL-18.1-fix2-9: Conditionality of Database Revision Tests [BTLWG-958, CR-0217]

Overview:
Upon revisiting CR-0217, the BTL-WG decided to change the conditionality of 7.3.2.10.X5 & X6.

Changes:

BTL Checklist Changes

[no changes]

BTL Test Plan Changes

[Modify entries in section 3.10.2, Device Object, Supports Database_Revision Property]

135.1-2013 - 7.3.2.10.5 - Successful Increment of the Database_Revision Property after Changing the Object_Name Property of an Object		
	Test Conditionality	If the device implements protocol revision 2 or higher, this test must be executed. ~~If the IUT does not support a changeable Object_Name property in any object, this test may be skipped.~~ *Where the only changeable Object_Name property in the device is the Device object's name, and the only method for changing the name is to replace the complete configuration and wipe all runtime data in the device, this test shall be skipped.*
	Test Directives	
	Testing Hints	
BTL - 7.3.2.10.6 - Successful Increment of the Database_Revision Property after Changing the Object_Identifier Property of an Object		
	Test Conditionality	If the device implements protocol revision 2 or higher, this test must be executed. ~~If the IUT does not support a changeable Object_Identifier property in any object, this test may be skipped.~~ *Where the only changeable Object_Identifier property in the device is the Device object's identifier, this test shall be skipped.*
	Test Directives	
	Testing Hints	

BTL Specified Test Changes

None

BTL-18.1-fix2-10: External Writes Test [BTLWG-967, CR-0262]

Overview:
CR-0262 was raised on this test135.1-2009 7.3.2.9.3.

Changes:

BTL Checklist Changes

[no changes]

BTL Test Plan Changes

[Modify Test Plan 3.9.4]

3.9.4 Supports External Writes

A Command object can write to external objects.

***BTL~~135.1-2013~~* - 7.3.2.9.3 - External Writes Test**		
	Test Conditionality	~~If the IUT does supports writing to external objects from a Command object, The test m~~Must be executed.
	Test Directives	
	Testing Hints	

BTL Specified Test Changes

[In BTL Specified Tests, move 131.2019 - 7.3.2.9.3 into BTL Specified Tests and modify as below:]

7.3.2.9.3 External Writes Test
Reason for Change: The purpose of the test is updated after removing the duplicate information. The Test Concept and Configuration Requirements are updated with generic test parameters.

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

~~BACnet Reference Clause: 12.10.8~~.

Purpose: To verify that a Command object can write to external objects. ~~If the IUT does not support writing to external objects from a Command object this test shall be omitted.~~

Test Concept: The IUT is configured with a Command object having an action list *X*, that includes writing to an *object O1* in the TD. The TD invokes this action list by writing the appropriate value to the Command object. The TD verifies that the IUT transmits the appropriate WriteProperty-Request.

Configuration Requirements: The IUT shall be configured with a Command object that has an Action property that contains an action list, X, that includes a command to write to the ~~Present_Value~~ *property P1* of ~~(Analog Value, 0)~~ *object O1 with the value V1* in the TD.

Note to Tester: Any WriteProperty request generated by the IUT may have a Priority parameter. If included, it shall be in the range 1-16, excluding 6.

Test Steps:

1. WRITE Present_Value = X
2. RECEIVE Simple-ACK-PDU
3. *VERIFY In_Process = TRUE*

~~*3.*~~ BEFORE **External Command Fail Time**
RECEIVE WriteProperty-Request,
'Object Identifier' = ~~(Analog Value, 0)~~ *O1*,
'Property Identifier' = ~~Present_Value~~ *P1*,
'Property Value' = ~~(any Real value)~~ *V1*
4. ~~VERIFY In_Process = TRUE~~TRANSMIT BACnet-SimpleACK-*PDU*
5. IF (Post_Delay is present)
WAIT (Post_Delay)
6. VERIFY In_Process = FALSE

~~Note to Tester: Any WriteProperty request generated by the IUT may have a Priority parameter. If included, it shall be in the range 1-16, excluding 6.~~

BTL-18.1-fix2-11: Review and Adjust BTL-R Entries [BTLWG-976]

Overview:
There are a number of BTL-R entries in the Checklist which the standard mandates. Most of these were changed at the request of the BTL. The Checklist should be updated to reflect that these are no longer BTL-R.

See original work item for rational for each change.

Changes:

BTL Checklist Changes

[Modify Data Sharing - Change Of Value - B section as shown]

Data Sharing - Change Of Value - B		
	R	Base Requirements
	R	Supports Lifetimes up to 8 hours in duration
	~~BTL~~-R[1]	Supports 5 concurrent COV subscribers
	…	
[1] *BTL-R if the IUT claims a revision before Protocol_Revision 4.*		

[Modify SCHED-E-B, to remove all references to BTL-R and add footnote]

Scheduling - External - B		
	R	Base Requirements
	R	Supports DS-WP-A
	R	Supports SCHED-I-B
	R	Supports writable List_Of_Object_Property_References property
	~~BTL~~-R[1]	Is able to schedule NULL values
	~~BTL~~-R[1]	Is able to schedule BOOLEAN values
	~~BTL~~-R[1]	Is able to schedule Unsigned values
	~~BTL~~-R[1]	Is able to schedule REAL values
	~~BTL~~-R[1]	Is able to schedule Enumerated values
[1] *BTL-R if IUT claims a revision before Protocol_Revision 20.*		

[Modify DM-TS-A and DM-UTC-A to remove the BTL-R per addenda ar]

Device Management - Time Synchronization - A		
	R	Base Requirements
	~~BTL~~-R[1]	Supports DM-UTC-A
[1] *BTL-R if IUT claims revision before Protocol_Revision 15.*		
Device Management - UTC Time Synchronization - A		
	R	Base Requirements
	~~BTL~~-R[1]	Supports DM-TS-A
[1] *BTL-R if IUT claims revision before Protocol_Revision 15.*		

[Modify DM-DCC-A to remove BTL-R for finite timeout selection. Add footnote.]

Device Management - Device Communication Control - A		
	R	Base Requirements
	BTL-R	Supports sending a DeviceCommunicationControl service request with an arbitrary password
	~~BTL~~-R[1]	Supports sending a DeviceCommunicationControl service request with a finite timeout
[1] *BTL-R if IUT claims revision before Protocol_Revision 12.*		

[Modify the Annex J - BBMD section to remove the BTL-R and add a footnote.]

BACnet/IP - Annex J - BBMD		
	R	Base Requirements
	R[1]	*Supports NM-BBMDC-B*
	~~BTL-~~R[2]	Supports a BDT with at least ~~four~~*five* entries
[1] *Required if the IUT claims Protocol_Revision 17 or later.* [2] *BTL-R if IUT claims revision before Protocol_Revision 17 and only require four entries.*		

[Change NM-FDR-A to remove the BTL-R and add footnote.]

Network Management - Foreign Device Registration - A		
	R	Base Requirements
	~~BTL-~~R[1]	Supports configurable BBMD Address
	…	
[1] *BTL-R if IUT claims revision before Protocol_Revision 17.*		

BTL Test Plan Changes

None

BTL Specified Test Changes

None

BTL-18.1-fix2-12: DM-RD-B Configuration Requirements for Negative Tests [BTLWG-996]

Overview:
The tests assume that the IUT is configured to require a password, but this is not stated by the test.

Changes:

BTL Checklist Changes

None

BTL Test Plan Changes

In Section 4.4.1 change references from 135.1-2019 to BTL:

***BTL~~135.1-2019~~* - 9.20.2.1 - Reading a Single, Unsupported Property from a Single Object**		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
***BTL~~135.1-2019~~* - 9.20.2.2 - Reading Multiple Properties with Access Errors for Every Property**		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

BTL Specified Test Changes

[From 135.1, modify 9.27.2.1, 9.27.2.2 and from BTL Specified Tests modify 9.27.2.3 and 9.27.2.4 by adding Configuration Requirements section to each]

Configuration Requirements: The IUT shall be configured to require a password for ReinitializeDevice.

BTL-18.1-fix2-13: Silenced Property Test does not Provide Enough Direction [BTLWG-1007, CR-0476]

Overview:
Test 7.3.2.15.X9 Silenced Property test does not provide any direction on the initial state of the life safety object nor does the Test Plan provide directives on how many or which configurations should be tested.

Changes:

BTL Checklist Changes

None

BTL Test Plan Changes

None

BTL Specified Test Changes

[In BTL Specified Tests 16.1, 7.3.2.15.X9 Silenced Property test, Page No. 91]

7.3.2.15.X9 Silenced Property test

Reason for Change: Added local matter test concept statement for LifeSafetyOperation service and Silienced state

Purpose: This test verifies the behavior of Silenced property.

Test Concept: Verify the interrelationship between the Silenced property and any audible or visual indication that has been silenced by the receipt of a LifeSafetyOperation service request or a local process. If the Silenced property of the object under test is unchanging by means of a LifeSafetyOperation service requests, because none of the silencing operations are supported, then this test shall be omitted. This test applies to Life Safety Zone and Life Safety Point object.

Since the result of any specific LifeSafetyOperation is a local matter, the expected actions when an operation is applied to an object is a local matter. In order to apply this test, the tester selects an initial Silenced state and a BACnetLifeSafetyOperation. The tester then verifies that the expected Silienced state, as specified by the vendor, is the result of the life safety operation on the object.

The tester will select one instance of each appropriate object type and test it as described.

Test Steps:
1. InitialSilencedState = READ Silenced
2. VERIFY Silenced = Other than InitialSilencedState
3. TRANSMIT LifeSafetyOperation-Request,
   'Requesting Process Identifier' = (any valid identifier),
   'Requesting Source' = (any valid character string),
   'Request' = (any supported LifeSafetyOperation request transmitted to silence the sounder/strobe),
   'Object Identifier' = (the selected object)
5. RECEIVE BACnet-SimpleACK-PDU
6. CHECK (Sounder/Strobe inactive)
7 ResultingSilencedState = READ Silenced
8. CHECK (the ResultingSilencedState is equal to the InitialSilencedState, modified by the LifeSafetyOperation request transmitted)

BTL-18.1-fix2-14: DS-COV-B and DS-COVP-B Test Changes [BTLWG-1021, CR-0484]

Overview:
As per BACnet Claus 13.14.2, if context doesn't match with earlier subscription then it will be considered as a new subscription. However, in the 135.1.2019 - 9.10.1.8 and 9.11.1.8 test cases test steps 1 and 4 do not match and yet it is considered as a re-subscription request of earlier subscription. Test steps also don't satisfy the purpose requirement.

Changes:

BTL Checklist Changes

None

BTL Test Plan Changes

[In BTL Test Plan, add new test to the Base Requirements of DS-COV-B]

4.10 Data Sharing - Change Of Value - B

4.10.1 Base Requirements

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

…		
BTL - 9.10.1.8 - Updating Existing Subscriptions		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

[In BTL Test Plan, change test reference in Base Requirements section of DS-COVP-B, as shown below]

4.20 Data Sharing - Change Of Value Property - B

4.20.1 Base Requirements

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

…		
BTL - 9.11.1.8 - Updating Existing Subscriptions		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

BTL Specified Test Changes

[In BTL Specified Tests, copy test 9.10.1.8 from 135-2019 and modify as shown]

9.10.1.8 Updating Existing Subscriptions

Reason for change: Modify the test case as per purpose, BACnet Claus 13.14.2, and updated the language as per BACnet 135.1: 6 Clause

Purpose: To verify that the IUT correctly responds to a SubscribeCOV request to update the lifetime of a subscription. Either confirmed or unconfirmed notifications may be used but at least one of these options must be supported by the IUT.

Test Concept: A subscription for COV notifications is made for 60 seconds. Before that subscription has expired a second subscription is made for 300 seconds. When the notification is sent in response to the second subscription the lifetime is checked to verify that it is greater than 60 but less than 300 seconds.

Test Steps:

1. TRANSMIT SubscribeCOV-Request,
   'Subscriber Process Identifier' = (*PID1*, any valid process identifier),
   'Monitored Object Identifier' = (*O1*, any object supporting COV notifications),
   'Issue Confirmed Notifications' = TRUE | FALSE,
   'Lifetime' = 60
2. RECEIVE BACnet-SimpleACK-PDU
3. IF (the subscription was for confirmed notifications) THEN
   BEFORE **Notification Fail Time**
   RECEIVE ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = *PID1*~~(the same identifier used in the subscription)~~,
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = *O1*~~(the same object used in the subscription)~~,
   'Time Remaining' = *(*60 *or less than 60)*,
   'List of Values' = (values appropriate to the object type of the monitored object)
   TRANSMIT BACnet-SimpleACK-PDU
   ELSE
   BEFORE **Notification Fail Time**
   RECEIVE UnconfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = *PID1*~~(the same identifier used in the subscription)~~,
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = *O1*~~(the same object used in the subscription)~~,
   'Time Remaining' = *(*~60, *but not greater than 60)*,
   'List of Values' = (values appropriate to the object type of the monitored object)
4. TRANSMIT SubscribeCOV-Request,
   'Subscriber Process Identifier' = *PID1*~~(any valid process identifier)~~,
   'Monitored Object Identifier' = *O1*~~(any object supporting COV notifications)~~,
   'Issue Confirmed Notifications' = TRUE | FALSE,
   'Lifetime' = (*T1,* a value between 180 and 300 seconds)
5. RECEIVE BACnet-SimpleACK-PDU
6. IF (the subscription was for confirmed notifications) THEN
   BEFORE **Notification Fail Time**
   RECEIVE ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = *PID1*~~(the same identifier used in the subscription)~~,
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = *O1*~~(the same object used in the subscription)~~,
   'Time Remaining' = *(~T1, but not greater than T1*~~the requested subscription lifetime~~),
   'List of Values' = (values appropriate to the object type of the monitored object)
   TRANSMIT BACnet-SimpleACK-PDU
   ELSE
   BEFORE **Notification Fail Time**
   RECEIVE UnconfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = *PID1*~~(the same identifier used in the subscription)~~,
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = *O1*~~(the same object used in the subscription)~~,
   'Time Remaining' = *(~T1, but not greater than T1*~~the requested subscription lifetime~~),
   'List of Values' = (values appropriate to the object type of the monitored object)

[In BTL Specified Tests, copy test 9.11.1.8 from 135-2019 and modify as shown]

9.11.1.8 Updating Existing Subscriptions

Reason for change: Modify the test case as per purpose, BACnet Claus 13.14.2, and updated the language as per BACnet 135.1: 6 Clause

Purpose: To verify that the IUT correctly responds to a SubscribeCOVProperty request to update the lifetime of a subscription. Either confirmed or unconfirmed notifications may be used but at least one of these options must be supported by the IUT.

Test Concept: A subscription for COV notifications is made for 60 seconds. Before that subscription has expired a second subscription is made for 300 seconds. When the notification is sent in response to the second subscription, the lifetime is checked to verify that it is greater than 60 but less than 300 seconds.

Test Steps:

1. TRANSMIT SubscribeCOVProperty-Request,
   'Subscriber Process Identifier' = (*PID1*, any valid process identifier),
   'Monitored Object Identifier' = (*O1*, any object supporting COV notifications),
   'Issue Confirmed Notifications' = TRUE | FALSE,
   'Lifetime' = 60
   'Monitored Property Identifier' = (*P1*, any valid property supporting COV notifications)
2. RECEIVE BACnet-SimpleACK-PDU
~~3. BEFORE Notification Fail Time~~
3. IF (the subscription was for confirmed notifications) THEN
   BEFORE Notification Fail Time
      RECEIVE BACnetConfirmedCOVNotification-Request,
         'Subscriber Process Identifier' = *PID1*~~(the same identifier used in the subscription)~~,
         'Initiating Device Identifier' = IUT,
         'Monitored Object Identifier' = *O1*~~(the same object used in the subscription)~~,
         'Time Remaining' = *(*60 *or less than 60)*,
         'List of Values' = (values appropriate to the object type *and subscribed to property* ~~of the monitored object including the changed value of that triggered the notification~~)
      TRANSMIT BACnet-SimpleACK-PDU
   ELSE
   BEFORE Notification Fail Time
      RECEIVE BACnetUnconfirmedCOVNotification-Request,
         'Subscriber Process Identifier' = *PID1*~~(the same identifier used in the subscription)~~,
         'Initiating Device Identifier' = IUT,
         'Monitored Object Identifier' = *O1*~~(the same object used in the subscription)~~,
         'Time Remaining' = *(*60 *or less than 60)*,
         'List of Values' = (values appropriate to the object type *and subscribed to property* ~~of the monitored object including the changed value of that triggered the notification~~)
4. TRANSMIT SubscribeCOVProperty-Request,
   'Subscriber Process Identifier' = *PID1*~~(any valid process identifier)~~,
   'Monitored Object Identifier' = O1~~(any object supporting COV notifications)~~,
   'Issue Confirmed Notifications' = TRUE | FALSE,
   'Lifetime' = (*T1,* a value between 180 and 300 seconds)
   'Monitored Property Identifier' = *P1*~~(any valid property supporting COV notifications)~~
5. RECEIVE BACnet-SimpleACK-PDU
~~6. BEFORE Notification Fail Time~~
6. IF (the subscription was for confirmed notifications) THEN
   BEFORE Notification Fail Time
      RECEIVE BACnetConfirmedCOVNotification-Request,
         'Subscriber Process Identifier' = *PID1*~~(the same identifier used in the subscription)~~,
         'Initiating Device Identifier' = IUT,
         'Monitored Object Identifier' = *O1*~~(the same object used in the subscription)~~,
         'Time Remaining' = (*~T1*, *but not greater than T1*~~the requested subscription lifetime~~),
         'List of Values' = (values appropriate to the object type *and subscribed to property* ~~of the monitored object including the changed value of that triggered the notification~~)
      TRANSMIT BACnet-SimpleACK-PDU
   ELSE
      BEFORE Notification Fail Time

RECEIVE BACnetUnconfirmedCOVNotification-Request,
'Subscriber Process Identifier' = *PID1* ~~(the same identifier used in the subscription)~~,
'Initiating Device Identifier' = IUT,
'Monitored Object Identifier' = *O1* ~~(the same object used in the subscription)~~,
'Time Remaining' = *(~T1, but not greater than T1* ~~the requested subscription lifetime~~),
'List of Values' = (values appropriate to the object type *and subscribed to property* ~~of the monitored object including the changed value of that triggered the notification~~)

BTL-18.1-fix2-15: Clarify Data Link Options in Checklist [BTLWG-1043]

Overview:
In each of the data links, there is an option 'Supports configuration through Network Port object', which is optional.
No descriptive paragraph is provided for this section to help in determining when it should be selected (which is pretty much always for devices with PR >= 17).

The checklist should be changed to make this option conditional, and a descriptive paragraph in the Test Plan should be added to the effect of: 'The IUT supports full, or partial, configuration of the data link through the Network Port object. Specifically, at least 1 property in the Network Port object which changes the behavior of the data link is writable.'

Changes:

Checklist Changes

[BTL Checklist 18.0; pages 47 – 48. Change each “Supports configuration through Network Port object” to C. Add footnote “Require for devices Protocol_Revision 17 or higher.]

9 Data Link Layer

Support	Listing	Option
Data Link Layer - MS/TP - Master Node		
	R	Base Requirements
	C[1]	Supports writable Max_Master property
	C[1]	Supports read only Max_Master property
	C[2]	Contains configurable Max_Info_Frames property
	C[2]	Contains non-configurable Max_Info_Frames property
	O	Is a BACnet router
	C[3,4]	Supports extended MS/TP frames (over 501 octets)
	~~O~~*C[5]*	Supports configuration through Network Port object
[1] Exactly one of these options is required in order to claim conformance to this BIBB. [2] Exactly one of these options is required in order to claim conformance to this BIBB. [3] Protocol_Revision 16 or higher must be claimed. [4] Required if the device is a router and claims Protocol_Revision 16 or higher. [5] *Required for devices which claim Protocol_Revision 17 or higher and which support DS-WP-B.*		
BACnet/IP - Annex J - non-BBMD Functionality		
	R	Base Requirements
	BTL-C[1]	Is able to register as a Foreign Device
	O	Is able to initiate Original-Broadcast-NPDU
	~~O~~*C[2]*	Supports configuration through Network Port object
	O	Supports Network Port objects and DHCP
[1] This option is required if the device does not claim Annex J BACnet/IP BBMD support. [2] *Required for devices which claim Protocol_Revision 17 or higher and the device supports FOREIGN mode.*		
BACnet/IP - Annex J - BBMD		
	R	Base Requirements
	BTL-R	Supports a BDT with at least four entries
	R	Registration by a Foreign Device is supported
	R	Supports 2-hop mode
	O	Supports 1-hop mode
	O	BBMD supports Network Address Translation
	O	Is able to initiate Original-Broadcast-NPDU
	~~O~~ *C[1]*	Supports configuration through Network Port object

Support	Listing	Option
	O	Supports Network Port objects and DHCP
[1] *Required for devices which claim Protocol_Revision 17 or higher.*		
Data Link Layer - IPv6		
	R	Base Requirements
	C[1]	Is able to operate in Normal mode
	C[1]	Is able to operate in Foreign mode
	C[2]	Is able to operate in BBMD mode
	~~*OR*~~	Supports configuration through Network Port object
	O	Supports DHCP
[1] Required if the device does not support BBMD mode. [2] Required if the device does not support Foreign mode.		

Test Plan Changes

[BTL Test Plan: Add introductory paragraph to each of the sections below]

[Test Plan 18.0 page 552]

9.1 Data Link Layer - MS/TP - Master Node

9.1.8 Supports Configuration Through Network Port Object

The IUT supports full, or partial, configuration of the data link through the Network Port object. Specifically, at least 1 property in the Network Port object which changes the behavior of the data link is writable.

BTL - 7.3.2.X62.1.1 - Configure Network Through Network Port Object Test		
	Test Conditionality	Must be executed.
	Test Directives	Perform at least once. Repeat each time the network is reconfigured for a test.
	Testing Hints	

[Test Plan 18.0 page 553]

9.2 Data Link Layer - MS/TP - Slave Node

9.2.2 Supports Configuration Through Network Port Object

The IUT supports full, or partial, configuration of the data link through the Network Port object. Specifically, at least 1 property in the Network Port object which changes the behavior of the data link is writable.

BTL - 7.3.2.X62.1.1 - Configure Network Through Network Port Object Test		
	Test Conditionality	Must be executed.
	Test Directives	Perform at least once. Repeat each time the network is reconfigured for a test.
	Testing Hints	

[Test Plan 18.0 page 557]

9.3 BACnet/IP - Annex J - non-BBMD Functionality

9.3.4 Supports Configuration Through Network Port Object

~~The IUT can be configured using the Network Port Object properties.~~ *The IUT supports full, or partial, configuration of the data link through the Network Port object. Specifically, at least 1 property in the Network Port object which changes the behavior of the data link is writable.*

BTL - 7.3.2.X62.1.1 - Configure Network Through Network Port Object Test		
	Test Conditionality	Must be executed.
	Test Directives	Perform at least once. Repeat each time the network is reconfigured for a test.
	Testing Hints	*The only required writable properties are for foreign devices so it may be that the IUT must first be placed into FOREIGN mode before this test is applied.*

[Test Plan 18.0 page 561]

9.4 BACnet/IP - Annex J - BBMD

9.4.8 Supports Configuration Through Network Port Object

The IUT supports full, or partial, configuration of the data link through the Network Port object. Specifically, at least 1 property in the Network Port object which changes the behavior of the data link is writable.

BTL - 7.3.2.X62.1.1 - Configure Network Through Network Port Object Test		
	Test Conditionality	Must be executed.
	Test Directives	Perform at least once. Repeat each time the network is reconfigured for a test.
	Testing Hints	

[Test Plan 18.0 page 562]

9.5 Data Link Layer - ZigBee

9.5.2 Supports Configuration Through Network Port Object

The IUT supports full, or partial, configuration of the data link through the Network Port object. Specifically, at least 1 property in the Network Port object which changes the behavior of the data link is writable.

BTL - 7.3.2.X62.1.1 - Configure Network Through Network Port Object Test		
	Test Conditionality	Must be executed.
	Test Directives	Perform at least once. Repeat each time the network is reconfigured for a test.
	Testing Hints	

[Test Plan 18.0 page 563]

9.6 Data Link Layer - Ethernet

9.6.2 Supports Configuration Through Network Port Object

The IUT supports full, or partial, configuration of the data link through the Network Port object. Specifically, at least 1 property in the Network Port object which changes the behavior of the data link is writable.

BTL - 7.3.2.X62.1.1 - Configure Network Through Network Port Object Test		
	Test Conditionality	Must be executed.
	Test Directives	Perform at least once. Repeat each time the network is reconfigured for a test.
	Testing Hints	

[Test Plan 18.0 page 564]

9.7 Data Link Layer - ARCNET

9.7.2 Supports Configuration Through Network Port Object

The IUT supports full, or partial, configuration of the data link through the Network Port object. Specifically, at least 1 property in the Network Port object which changes the behavior of the data link is writable.

BTL - 7.3.2.X62.1.1 - Configure Network Through Network Port Object Test		
	Test Conditionality	Must be executed.
	Test Directives	Perform at least once. Repeat each time the network is reconfigured for a test.
	Testing Hints	

[Test Plan 18.0 page 565]

9.8 Data Link Layer - LonTalk

9.8.2 Supports Configuration Through Network Port Object

The IUT supports full, or partial, configuration of the data link through the Network Port object. Specifically, at least 1 property in the Network Port object which changes the behavior of the data link is writable.

BTL - 7.3.2.X62.1.1 - Configure Network Through Network Port Object Test		
	Test Conditionality	Must be executed.
	Test Directives	Perform at least once. Repeat each time the network is reconfigured for a test.
	Testing Hints	

[Test Plan 18.0 page 569]

9.9 Data Link Layer - IPv6

9.9.5 Supports Configuration Through Network Port Object

The IUT supports full, or partial, configuration of the data link through the Network Port object. Specifically, at least 1 property in the Network Port object which changes the behavior of the data link is writable.

BTL - 7.3.2.X62.1.1 - Configure Network Through Network Port Object Test		
	Test Conditionality	Must be executed.
	Test Directives	Perform at least once. Repeat each time the network is reconfigured for a test.
	Testing Hints	

[Test Plan 18.0 page 576]

9.10 Data Link Layer - Secure Connect

9.10.7 Supports Configuration Through Network Port Object

The IUT supports full, or partial, configuration of the data link through the Network Port object. Specifically, at least 1 property in the Network Port object which changes the behavior of the data link is writable.

BTL - 7.3.2.X62.1.1 - Configure Network Through Network Port Object Test		
	Test Conditionality	Must be executed.
	Test Directives	Perform at least once. Repeat each time the network is reconfigured for a test.
	Testing Hints	

BTL Specified Test Changes

None

BTL-18.1-fix2-16: Resizing a Writable Fixed Array Property Test Changes [BTLWG-1069, CR-0488]

Overview:
In response to CR-0488 the BTLWG agreed that WRITE_ACCESS_DENIED should be an acceptable error code in some cases in test 9.22.2.X2

Changes:

BTL Checklist Changes

None

BTL Test Plan Changes

None

BTL Specified Test Changes

9.22.2.X2 Resizing a writable fixed size array property

Reason for Change: No test exists for this functionality.

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

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

Test Steps:

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

BTL-18.1-fix2-17: Internal Processing Fail Time [BTLWG-1085]

Overview:
Internal Processing Fail Time should note that it is a maximum and that it should be allowed to be specified on a per test basis to allow tests to run faster.

Changes:

BTL Test Plan Changes

none

BTL Test Plan Changes

none

BTL Specified Test Changes

[Move clause 6.3.2 from 135.1-2019 into BTL Specified Tests, and modify]

6.3.2 Internal Processing Fail Time

The Internal Processing Fail Time is the elapsed time, in seconds, between the receipt of a write to a BACnet property or some other event that changes the value of the property and when a test is considered to have failed because the property value has not been updated.

The Internal Processing Fail Time contained in the EPICS is the maximum value for all situations. In order to reduce test time, it is acceptable that a shorter Internal Processing Fail Time value be used on a per test basis where the shorter time will not negatively impact the test result.

BTL-18.1-fix2-18: Load Control Level Test Fixes [BTLWG-1091, CR-0494]

Overview:
In CR-0494, it is noted that there are a number of issues with test 7.3.2.X53.1 dealing with the initial value for Start_Time.

Changes:

BTL Checklist Changes

None

BTL Test Plan Changes

None

BTL Specified Tests Changes

[In BTL Specified Tests, modify test 7.3.2.X53.1]

7.3.2.X53.1Requested_Shed_Level property test with LEVEL choice
Reason for Change: This test is not specified in any SSPC proposal.

Purpose: To verify that the IUT can accept and execute a shed request with LEVEL choice.

Test Concept: The Requested_Shed_Level property of the Load Control object is set to a LEVEL choice and it is verified that the Load Control object behaves as per the Load Control state machine.

Configuration Requirements: The IUT shall be configured so that Present_Value is equal to SHED_INACTIVE, at the start of the test. The start time, ST, and shed duration, SD, shall be selected such that there is at least 1 duty window, DW, of time between the current time and ST + SD.

Notes to Tester: The writing of Duty_Window can be skipped, for the tester to see that the VERIFY Duty_Window = DW during a pending or active shed event, that property takes on PAV, the configured pre-agreed upon value.

Test Steps:
1. VERIFY Requested_Shed_Level = (DRSL: one of the default Requested_Shed_Level values for a previous shed request, not necessarily the LEVEL default of 0)
2. VERIFY Expected_Shed_Level = DRSL
3. VERIFY Actual_Shed_Level = DRSL
4. VERIFY Present_Value = SHED_INACTIVE
5. VERIFY Shed_Duration = 0
6. VERIFY Start_Time = (the fully unspecified datetime value)
7. VERIFY Duty_Window = (PAV, the pre-agreed upon value)
8. WRITE Enable = TRUE
9. WRITE Shed_Duration = (SD, any value appropriate to the object)
10. WRITE Start_Time = (ST, any valid start time including values in the past, present or future, but limited such that current_time is before ST + SD)
11. WRITE Duty_Window = (DW, any value appropriate to the object)
12. WRITE Requested_Shed_Level = (a value appropriate to the object with a LEVEL choice, that is not equal to the default value: 0)
13. IF (current time is before ST) THEN
        VERIFY Present_Value = (SHED_REQUEST_PENDING,
                                SHED_COMPLIANT or
                                SHED_NON_COMPLIANT)
    WAIT (until Start_Time)
14. VERIFY Present_Value = (SHED_COMPLIANT or SHED_NONCOMPLIANT)
15. IF (ST + DW < ST + SD and ST + DW is in the future) THEN

WAIT (until ST+DW)
16. IF (current time is after ST+DW) THEN
IF (Actual_Shed_Level does not comply with Requested_Shed_Value) THEN
VERIFY Present_Value = SHED_NONCOMPLIANT
17. VERIFY Shed_Duration = SD
18. VERIFY Start_Time = ST
19. VERIFY Duty_Window = DW
20. VERIFY Expected_Shed_Level = (any value appropriate to the choice, that is not equal to the default value)
21. VERIFY Actual_Shed_Level = (any value appropriate to the choice, that is not equal to the default value)
-- the above VERIFY statements apply all through the time that there is a pending or active shed event
22. WAIT (until the shed request has completed, at ST+SD)
23. VERIFY Requested_Shed_Level = 0 -- the default LEVEL value
24. VERIFY Expected_Shed_Level = 0 -- the default LEVEL value
25. VERIFY Actual_Shed_Level = 0 -- the default LEVEL value
26. VERIFY Shed_Duration = 0
27. VERIFY Start_Time = (the fully unspecified datetime value)
28. VERIFY Duty_Window = PAV