BACnet® TESTING LABORATORIES ADDENDA

Addendum f to BTL Test Package 15.1

Revision 3.0
Revised 9/06/2018

Approved by the BTL Working Group Voting Members on 06-Sept-2018;
Published on 06-Sept-2018

[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-15.1f-1: Binary Object Elapsed Active Time Changes**, p. 2** [wID0432, BTLWG-39]
BTL-15.1f-2: File Object, **p. 3** [BTLWG-373, Interim Item]
BTL-15.1f-3: Read Only Recipient List Test Considerations**, p. 5** [BTLWG-392, Interim Item]
BTL-15.1f-4: Allow Unicast I-Have**, p. 7** [BTLWG-170, wID0352]
BTL-15.1f-5: Update COV Tests, **p. 19** [BTLWG-166, wID0940]
BTL-15.1f-6: Allow Devices to Cancel COV, **p. 24** [BTLWG-350]
BTL-15.1f-7: ReadRange Support for ALL List properties, **p. 26** [BTLWG-398]

In the following document, language to be added to existing clauses within the BTL Test Package 15.1 is indicated through the use of *italics*, while deletions are indicated by ~~strikethrough~~. Where entirely new subclauses are proposed to be added, plain type is used throughout

In addition, changes to BTL Specified Tests also contain a yellow highlight to indicate the changes made by this addendum.

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

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

BTL-15.1f-1: Binary Object Elapsed Active Time Changes

Overview:

Addendum 135-2010ad-3 added the alternative to allow Feedback_Value to be used for Elapsed_Active_Time tracking. This document makes needed changes in the BTL Test Package to provide for that.

These changes are not contained in any SSPC proposal.

Changes:

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

7.3.1.9 - Binary Object Elapsed Active Time Tests
Reason for Change: Errors were pointed out via BTL-CR-0253. Further changes are applied for 135-2010ad-3.

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

~~BACnet Reference Clauses: 12.6.17, 12.6.18, 12.7.17, 12.7.18, 12.8.15, and 12.8.16.~~

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

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

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

Notes To Tester: It was intentional to specify that the alternative use of Feedback_Value tracking specified in 135-2010ad-3 is allowed regardless of the Protocol_Revision claimed by the implementation.

Test Steps: <No changes made>

BTL-15.1f-2: File Object

Overview:
The uses of File for a purpose other than Backup and Restore, all seem to be essentially proprietary. We've seen write-only Files. We've seen Files that cannot to resized. We've seen Files that cannot be transferred except all at once. All existing tests in the 9.12 and 9.13 sections of 135.1-2013 for AtomicReadFile and AtomicWriteFile will only succeed for a small and overly-simplistic implementation of Files.

Every possible behavior of File and every possible set of parameter combinations in AtomicReadFile and AtomicWriteFile which will succeed, or what response they will send for failures, all seem to pose no interoperability issues.

Changes:

[In BTL Checklist, add/update File object type to Section 3, Objects]

Support	Listing	Option
. . .		
File Object		
	R	Base Requirements
	C[1]	Supports DM-BR-B
	C[1]	Supports a File object for a purpose other than Backup and Restore
	O	Contains a writable File for a purpose other than Backup and Restore
[1] At least one of these options is required if the IUT supports the File object type.		

[In BTL Test Plan, add File object tests in section 3.X48]

3.X48 File Object

3.X48.1 Base Requirements

For File object, there are no base requirements.

3.X48.2 Supports DM-BR-B

The IUT supports a data File that is readable and writable during Backup and Restore using AtomicReadFile and AtomicWriteFile requests.

Verify Checklist		
	Test Method	Manual
	Configuration	
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for Device Management - Backup and Restore - B in the Checklist.
	Testing Hints	
	Notes & Results	

3.X48.3 Supports a File object for a purpose other than Backup and Restore

For a device which contains a File object for a purpose other than Backup and Restore, there are no testing requirements.

3.X48.4 Contains a writable File for a purpose other than Backup and Restore

For a device which contains a writable File object for a purpose other than Backup and Restore, there are no testing requirements.

BTL-15.1f-3: Read Only Recipient List Test Considerations

Overview:

Device claiming it has a read-only Recipient_List property in a Notification class object must claim Protocol_Revision 13 or higher and must comply with the following section.

From 135-2010af-30:
Rationale
In order to reduce the level of effort required to implement basic event reporting, the notification distribution requirements are reduced. This change works in concert with the addition of the Notification Forwarder object that is added in a different section of this addendum. This change also allows a large number of Notification_Class objects to share a small number of sets of recipients by allowing a Notitication_Ciass to defer the recipient information to local Notification Forwarder objects.

Changes:
Add options to the BTL Checklist, BTL Test Plan and added new tests for coverage.

[In BTL Checklist, in the **Notification Class Object** revise conformance code, and add indicated line item.]

Notification Class Object		
	R	Base Requirements
	BTL-R	Supports DM-DDB-A
	~~BTL-R~~*C*[1]	Supports writable Recipient_List properties
	C[1]	*Supports read-only Recipient_List properties*
[1] *At least one of these options must be supported.*		

[In BTL Test Plan, add a new section under Notification Class Object for Supports read-only Recipient_List Properties.]

3.17 Notification Class Object

3.17.4 Supports read-only Recipient_List Properties

The IUT supports read-only Recipient_List properties.

BTL - 7.3.2.21.3.X9 Read-only Recipient_List for external Notification Forwarder objects		
	Test Method	
	Configuration	As per ***BTL Specified Tests.***
	Test Conditionality	Must be executed if the IUT does not claim support for Notification Forwarder objects.
	Test Directives	
	Testing Hints	
	Notes & Results	

[In BTL Test Plan, modify existing Base Requirements section under Alarm and Event - Notification - Internal-B. Modified sections. Changes are yellow highlighted for clarity.]

5.2 Alarm and Event - Notification - Internal - B

5.2.1 Base Requirements

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

BTL - 7.3.1.10 - Event_Enable Tests		
	Test Method	Manual

	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If the IUT cannot be configured to meet the configuration requirements then this test shall be skipped.
	Test Directives	If Event Enrollment objects are supported, ensure this functionality is tested on Event Enrollment objects.
	Testing Hints	The BTL will apply this to a single object. The pretester should apply it to all objects that support alarm generation.
	Notes & Results	
BTL - 7.3.1.12 - Notify_Type Test		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests***.
	Test Conditionality	If the IUT cannot be configured to meet the 135.1-2013 configuration requirements then this test shall be skipped.
	Test Directives	If Event Enrollment objects are supported, ensure this functionality is also tested on Event Enrollment objects.
	Testing Hints	
	Notes & Results	
135.1-2009 - 8.4 - ConfirmedEventNotification Service Initiation Tests		
	Test Method	
	Configuration	As per ***ASHRAE 135.1-2013***
	Test Conditionality	Must be executed unless IUT only supports read-only Recipient_List properties and does not claim Notification Forwarder objects. Any of the 8.4 tests can be used to ensure that the IUT properly generates ConfirmedEventNotification requests. The specific tests that can be executed are detailed under the test cases for the specific algorithms. As long as one of the tests is executed using ConfirmedEventNotifications, then this test case shall be satisfied.
	Test Directives	
	Testing Hints	
	Notes & Results	
135.1-2009 - 8.5 - UnconfirmedEventNotification Service Initiation Tests		
	Test Method	
	Configuration	
	Test Conditionality	Must be executed. Any of the 8.5 tests can be used to ensure that the IUT properly generates UnconfirmedEventNotification requests. The specific tests that can be executed are detailed under the test cases for the specific algorithms. As long as one of the tests is executed using UnconfirmedEventNotifications, then this test case shall be satisfied.
	Test Directives	
	Testing Hints	
	Notes & Results	

BTL-15.1f-4: Allow Unicast I-Have

Overview:

The BACnet standard (as per addendum 135-2012ar-5) now allows the IUT to send and receive a unicast response for IUT's claiming Protocol Revision equal or greater than 15.

[In BTL Specified Tests, Change tests 9.32.1.1 through 9.32.1.11as shown]

9.32.1.1 Object ID Version with No Device Range

Reason for Change: Modified test to remove dependency on EPICS values.

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 ~~**Internal Processing Fail Time**~~ ***Unconfirmed Response Fail Time***
4. *IF (Protocol_Revision is present and Protocol_Revision >= 15) THEN*
   RECEIVE DA = LOCAL BROADCAST | GLOBAL BROADCAST | TD,
   I-Have-Request,
   'Device Identifier' = (the IUT's Device object),
   'Object Identifier' = Object1
   'Object Name' = V1
   ELSE
   RECEIVE
   DA = LOCAL BROADCAST | GLOBAL BROADCAST,
   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.

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 ~~**Internal Processing Fail Time**~~ ***Unconfirmed Response Fail Time***
4. *IF (Protocol_Revision is present and Protocol_Revision >= 15) THEN*

RECEIVE DA = LOCAL BROADCAST | GLOBAL BROADCAST | TD,
I-Have-Request,
'Device Identifier' = (the IUT's Device object),
'Object Identifier' = Object1
'Object Name' = V1
ELSE
RECEIVE
DA = LOCAL BROADCAST | GLOBAL BROADCAST,
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.

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 ~~**Internal Processing Fail Time**~~ ***Unconfirmed Response Fail Time***
4. *IF (Protocol_Revision is present and Protocol_Revision >= 15) THEN*
   RECEIVE DA = LOCAL BROADCAST | GLOBAL BROADCAST | TD,
   I-Have-Request,
   'Device Identifier' = (the IUT's Device object),
   'Object Identifier' = Object1
   'Object Name' = V1
   ELSE
   RECEIVE
   DA = LOCAL BROADCAST | GLOBAL BROADCAST,
   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)~~

9.32.1.5 Object Name Version with IUT Inside of the Device Range

Reason for Change: Modified test to remove dependency on EPICS values.

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 ~~**Internal Processing Fail Time**~~ ***Unconfirmed Response Fail Time***
4. *IF (Protocol_Revision is present and Protocol_Revision >= 15) THEN*
   RECEIVE DA = LOCAL BROADCAST | GLOBAL BROADCAST | *TD,*
   I-Have-Request,
   'Device Identifier' = (the IUT's Device object),
   'Object Identifier' = Object1
   'Object Name' = V1
   ELSE
   RECEIVE
   DA = LOCAL BROADCAST | GLOBAL BROADCAST,
   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.

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 ~~**Internal Processing Fail Time**~~ ***Unconfirmed Response Fail Time***
4. *IF (Protocol_Revision is present and Protocol_Revision >= 15) THEN*
   RECEIVE DA = LOCAL BROADCAST | GLOBAL BROADCAST | TD
   I-Have-Request,
   'Device Identifier' = (the IUT's Device object),
   'Object Identifier' = Object1
   'Object Name' = V1
   ELSE
   RECEIVE
   DA = LOCAL BROADCAST | GLOBAL BROADCAST,
   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.

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 ~~**Internal Processing Fail Time**~~ ***Unconfirmed Response Fail Time***
4. *IF (Protocol_Revision is present and Protocol_Revision >= 15) THEN*
   RECEIVE DA = LOCAL BROADCAST | GLOBAL BROADCAST | TD,
   I-Have-Request,
   'Device Identifier' = (the IUT's Device object),
   'Object Identifier' = Object1
   'Object Name' = V1
   ELSE
   RECEIVE
   DA = LOCAL BROADCAST | GLOBAL BROADCAST,
   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.

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 ~~**Internal Processing Fail Time**~~ ***Unconfirmed Response Fail Time***
4. *IF (Protocol_Revision is present and Protocol_Revision >= 15) THEN*
   RECEIVE DA = LOCAL BROADCAST | GLOBAL BROADCAST | TD,
   I-Have-Request,
   'Device Identifier' = (the IUT's Device object),
   'Object Identifier' = Object1
   'Object Name' = V1
   ELSE
   RECEIVE
   DA = LOCAL BROADCAST | GLOBAL BROADCAST,
   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.

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 ~~**Internal Processing Fail Time**~~ ***Unconfirmed Response Fail Time***
4. *IF (Protocol_Revision is present and Protocol_Revision >= 15) THEN*
   RECEIVE DA = LOCAL BROADCAST | GLOBAL BROADCAST | TD,
   I-Have-Request,
   'Device Identifier' = (the IUT's Device object),
   'Object Identifier' = Object1
   'Object Name' = V1
   ELSE
   RECEIVE
   DA = LOCAL BROADCAST | GLOBAL BROADCAST,
   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)~~

9.32.1.11 Object Name Version, Directed to a Specific MAC Address

Reason for Change: Modified test to remove dependency on EPICS values.

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 ~~**Internal Processing Fail Time**~~ ***Unconfirmed Response Fail Time***
4. *IF (Protocol_Revision is present and Protocol_Revision >= 15) THEN*
   RECEIVE DA = LOCAL BROADCAST | GLOBAL BROADCAST | TD,
   I-Have-Request,
   'Device Identifier' = (the IUT's Device object),
   'Object Identifier' = Object1
   'Object Name' = V1

ELSE
RECEIVE
DA = LOCAL BROADCAST | GLOBAL BROADCAST,
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)~~

[In BTL Specified Tests, add new test derived from the version in 135.1-2013, as shown]

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 for IUT's claiming Protocol Revision equal or greater than 15.

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. *IF (Protocol_Revision is present and Protocol_Revision >= 15) THEN*
   RECEIVE DA = LOCAL BROADCAST | GLOBAL BROADCAST | TD,
   I-Have-Request,
   'Device Identifier' = (the IUT's Device object),
   'Object Identifier' = O1,
   'Object Name' = V2
   ELSE
   RECEIVE DA = LOCAL BROADCAST | GLOBAL BROADCAST
   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 for IUT's claiming Protocol Revision equal or greater than 15.

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. *IF (Protocol_Revision is present and Protocol_Revision >= 15) THEN*
   RECEIVE DA = LOCAL BROADCAST | GLOBAL BROADCAST | TD,
   I-Have-Request,
   'Device Identifier' = (the IUT's Device object),
   'Object Identifier' = O2
   ELSE
   RECEIVE DA = LOCAL BROADCAST | GLOBAL BROADCAST
   I-Have-Request,
   'Device Identifier' = (the IUT's Device object),
   'Object Identifier' = O2

[In BTL Specified Tests, modify tests as shown below.]

9.32.2.1 Object ID Version, Global Broadcast from a Remote Network

Reason for Change: Modified test to remove dependency on EPICS values.

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 ~~**Internal Processing Fail Time**~~ ***Unconfirmed Response Fail Time***
4. *IF (Protocol_Revision is present and Protocol_Revision >= 15) THEN*
RECEIVE
DESTINATION = GLOBAL BROADCAST | REMOTE BROADCAST (to the network X) | TD (DNET = X, DADR = Y),
I-Have-Request,
'Device Identifier' = (the IUT's Device object),
'Object Identifier' = Object1,
'Object Name' = V1
ELSE
RECEIVE
DESTINATION = GLOBAL BROADCAST | REMOTE BROADCAST (to the network *X* ~~specified in step 1~~),
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.

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 = (*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 ~~**Internal Processing Fail Time**~~ ***Unconfirmed Response Fail Time***
4. *IF (Protocol_Revision is present and Protocol_Revision >= 15) THEN*
RECEIVE
DA = GLOBAL BROADCAST | REMOTE BROADCAST (to the network X) | TD (DNET = X, DADR = Y),
I-Have-Request,
'Device Identifier' = (the IUT's Device object),
'Object Identifier' = Object1,
'Object Name' = V1
ELSE
RECEIVE
DA = GLOBAL BROADCAST | REMOTE BROADCAST (to the network *X* ~~specified in step 1~~)
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)~~

[In BTL Test Plan, Change the reference from 135.1-2013 to BTL]

~~135.1-2013~~ BTL - 9.32.1.12 - Who-Has After Object_Name Changed		
	Test Method	Manual
	Configuration	As per ~~ASHRAE 135.1-2013~~ **BTL Specified Tests.**
	Test Conditionality	If the IUT contains an object whose Object_Name can be changed, then this test shall be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

~~135.1-2013~~ BTL - 9.32.1.13 - Who-Has After Object_Identifier Changed		
	Test Method	Manual
	Configuration	As per ~~ASHRAE 135.1-2013~~ **BTL Specified Tests.**
	Test Conditionality	If the IUT contains an object whose Object_Identifier can be changed, then this test shall be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

[In BTL Specified Tests, Add the following tests based on the existing tests in 135.1.]

8.32 Who-Has Service Initiation Tests

8.32.1 Object Identifier Selection with no Device Instance Range

Reason for Change: The BACnet standard (per addendum 135-2012ar-5) now allows the IUT to send and receive a unicast response for IUT's claiming Protocol Revision equal or greater than 15.

Purpose: To verify that the IUT can initiate Who-Has service requests using the object identifier form with no device instance range. If the IUT cannot be caused to issue a Who-Has request of this form, then this test shall be omitted.

Test Steps:

1. RECEIVE
   DESTINATION = LOCAL BROADCAST | GLOBAL BROADCAST
   SOURCE = IUT,
   Who-Has-Request,
   'Object Identifier' = *Object1* ~~(any object identifier)~~
2. *IF (Protocol_Revision is present and Protocol_Revision >= 15) THEN*
   TRANSMIT
   DESTINATION = IUT | LOCAL BROADCAST | GLOBAL BROADCAST
   SOURCE = TD,
   I-Have-Request,
   'Device Identifier' = (the TD's Device object)
   'Object Identifier' = Object1
   ELSE
   TRANSMIT
   DESTINATION = LOCAL BROADCAST | GLOBAL BROADCAST
   SOURCE = TD,
   I-Have-Request,
   'Device Identifier' = (the TD's Device object)
   'Object Identifier' = Object1
3. *CHECK (for any vendor-defined observable actions)*

Notes to Tester: If there is no vendor-defined observable action, then test step 3 can be skipped.

8.32.2 Object Name Selection with no Device Instance Range

Reason for Change: The BACnet standard (per addendum 135-2012ar-5) now allows the IUT to send and receive a unicast response for IUT's claiming Protocol Revision equal or greater than 15.

Purpose: To verify that the IUT can initiate Who-Has service requests using the object name form with no device instance range. If the IUT cannot be caused to issue a Who-Has request of this form, then this test shall be omitted.

Test Steps:
1. RECEIVE
   DESTINATION = LOCAL BROADCAST | GLOBAL BROADCAST,
   SOURCE = IUT,
   Who-Has-Request,
   'Object Name' = *V1* ~~(any CharacterString)~~
2. *IF (Protocol_Revision is present and Protocol_Revision >= 15) THEN*
   TRANSMIT
   DESTINATION = IUT | LOCAL BROADCAST | GLOBAL BROADCAST
   SOURCE = TD,
   I-Have-Request,
   'Device Identifier' = (the TD's Device object)
   'Object Name' = V1
   ELSE
   TRANSMIT
   DESTINATION = LOCAL BROADCAST | GLOBAL BROADCAST
   SOURCE = TD,
   I-Have-Request,
   'Device Identifier' = (the TD's Device object)
   'Object Name' = V1
3. *CHECK (for any vendor-defined observable actions)*

Notes to Tester: If there is no vendor-defined observable action, then test step 3 can be skipped.

[In BTL Specified Tests, modify the existing tests with the changes highlighted below]

8.32.3 Object Identifier Selection with a Device Instance Range

Reason for Change: The allowed device instance range is from 0 - 4194303 and is specified in sections 16.9.1.1.1 and 16.10.1.1.1. The corresponding tests incorrectly set the low limit to 1. The BACnet standard (per addendum 135-2012ar-5) now allows the IUT to send and receive a unicast response for IUT's claiming Protocol Revision equal or greater than 15.

Purpose: To verify that the IUT can initiate Who-Has service requests using the object identifier form with a device instance range. If the IUT cannot be caused to issue a Who-Has request of this form, then this test shall be omitted.

Test Steps:
1. RECEIVE
   DESTINATION = LOCAL BROADCAST | GLOBAL BROADCAST,
   SOURCE = IUT,
   Who-Has-Request,
   'Device Instance Range Low Limit' = (any integer X: ~~1~~0 <= X <= 'Device Instance Range High Limit'),
   'Device Instance Range High Limit' = (any integer Y: 'Device Instance Range Low Limit' <= Y <= 4,194,303),
   'Object Identifier' = *Object1* ~~(any object identifier)~~
2. *IF (Protocol_Revision is present and Protocol_Revision >= 15) THEN*
   TRANSMIT
   DESTINATION = IUT | LOCAL BROADCAST | GLOBAL BROADCAST
   SOURCE = TD,
   I-Have-Request,
   'Device Identifier' = (the TD's Device object)
   'Object Identifier' = Object1
   ELSE
   TRANSMIT
   DESTINATION = LOCAL BROADCAST | GLOBAL BROADCAST
   SOURCE = TD,
   I-Have-Request,
   'Device Identifier' = (the TD's Device object)
   'Object Identifier' = Object1
3. *CHECK (for any vendor-defined observable actions)*

Notes to Tester: Device instance range should be selected to cover TD's device object identifier. If there is no vendor-defined observable action, then test step 3 can be skipped.

8.32.4 Object Name Selection with a Device Instance Range

Reason for Change: The allowed device instance range is from 0 - 4194303 and is specified in sections 16.9.1.1.1 and 16.10.1.1.1. The corresponding tests incorrectly set the low limit to 1. The BACnet standard (per addendum 135-2012ar-5) now allows the IUT to send and receive a unicast response for IUT's claiming Protocol Revision equal or greater than 15.

Purpose: To verify that the IUT can initiate Who-Has service requests using the object name form with a device instance range. If the IUT cannot be caused to issue a Who-Has request of this form, then this test shall be omitted.

Test Steps:

1. RECEIVE
   DESTINATION = LOCAL BROADCAST | GLOBAL BROADCAST,
   SOURCE = IUT,
   Who-Has-Request,
   'Device Instance Range Low Limit' = (any integer X: ~~1~~0 <= X <= 'Device Instance Range High Limit'),
   'Device Instance Range High Limit' = (any integer Y: 'Device Instance Range Low Limit' <= Y <= 4,194,303),
   'Object Name' = *V1* ~~(any CharacterString)~~
2. *IF (Protocol_Revision is present and Protocol_Revision >= 15) THEN*
   TRANSMIT
   DESTINATION = IUT | LOCAL BROADCAST | GLOBAL BROADCAST
   SOURCE = TD,
   I-Have-Request,
   'Device Identifier' = (the TD's Device object)
   'Object Name' = V1
   ELSE
   TRANSMIT
   DESTINATION = LOCAL BROADCAST | GLOBAL BROADCAST
   SOURCE = TD,
   I-Have-Request,
   'Device Identifier' = (the TD's Device object)
   'Object Name' = V1
3. *CHECK (for any vendor-defined observable actions)*

Notes to Tester: Device instance range should be selected to cover TD's device object identifier. If there is no vendor-defined observable action, then test step 3 can be skipped.

[In BTL Test Plan, Change test reference to new BTL Test from ASHRAE Tests]

8.3.1 Base Requirements

Base requirements must be met by any IUT claiming conformance to this BIBB. There are no base requirements tests for this section.

8.3.2 Initiates Who-Has Service Request with Object Identifier Parameter and no Range Parameters

The IUT can initiate a Who-Has service request with an ObjectID and No Range Parameters.

~~135.1-2013~~ *BTL* - 8.32.1 - Object Identifier Selection with no Device Instance Range		
	Test Method	Manual
	Configuration	As per ~~**ASHRAE 135.1-2013** ***BTL Specified Tests***~~**.**
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

8.3.3 Initiates Who-Has Service Request with Object Identifier Parameter with Range Parameters

The IUT can initiate a Who-Has service request with an ObjectID and Range Parameters.

BTL - 8.32.3 - Object Identifier Selection with a Device Instance Range		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests*.**
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

8.3.4 Initiates Who-Has Service Request with Object Name Parameter and no Range Parameters

The IUT can initiate a Who-Has service request with an Object Name and No Range Parameters.

~~135.1-2013~~ *BTL* - 8.32.2 - Object Name Selection with no Device Instance Range		
	Test Method	Manual
	Configuration	As per **~~ASHRAE 135.1-2013~~ *BTL Specified Tests*.**
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

8.3.5 Initiates Who-Has Service Request with Object Name Parameter with Range Parameters

The IUT can initiate a Who-Has service request with an Object Name and Range Parameters.

BTL - 8.32.4 - Object Name Selection with a Device Instance Range		
	Test Method	Manual
	Configuration	As per ***BTL Specified Tests*.**
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
	Notes & Results	

BTL-15.1f-5: Update COV Tests

Overview:

Some COV tests make reference to the COV_Increment property in the configuration requirements where it does not apply. These tests also do not make allowances for the case where Status_Flags may change during the test so the wording has been corrected to reflect 'Current Status_Flags'.

Changes:

[In BTL Specified Tests, change tests 8.2.3, 8.2.4, and 8.2.5 as shown]

8.2.3 Change of Value Notification from a Binary Input, Binary Output, and Binary Value Object Present_Value Property

Reason for Change: Updated the 'Configuration Requirements'. Removed extraneous SimpleACKs that appear after WRITE statements. Modified descriptive text for 'List of Values' properties.

Purpose: To verify that the IUT can initiate ConfirmedCOVNotification service requests conveying a change of the Present_Value property of Binary Input, Binary Output, and Binary Value objects.

Test Concept: A subscription for COV notifications is established, using a Lifetime of L. L shall be set to a value less than 24 hours and large enough to complete the test. The Present_Value of the monitored object is changed and a notification shall be received. The Present_Value may be changed using the WriteProperty service or by another means such as changing the input signal represented by a Binary Input object. For some implementations it may be necessary to write to the Out_Of_Service property first to accomplish this task. For implementations where it is not possible to write to these properties at all the vendor shall provide an alternative trigger mechanism to accomplish this task. All of these methods are equally acceptable.

Configuration Requirements: At the beginning of the test, the Out_Of_Service property shall have a value of FALSE. *Select an object where Present_Value is not expected to change outside the tester's control ~~by more than COV_Increment~~ or which has a writable Out_Of_Service.*

Test Steps:

REPEAT X = (one supported object of each type from the set Binary Input, Binary Output, and Binary Value) DO {

1. TRANSMIT SubscribeCOV-Request,
   'Subscriber Process Identifier' = (any value > 0 chosen by the TD),
   'Monitored Object Identifier' = X,
   'Issue Confirmed Notifications' = TRUE,
   'Lifetime' = L
2. RECEIVE BACnet-SimpleACK-PDU
3. BEFORE **Notification Fail Time**
   RECEIVE ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = (the same value used in step 1),
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = X,
   'Time Remaining' = (any value appropriate for the Lifetime selected),
   'List of Values' = (the initial Present_Value and initial Status_Flags)
4. TRANSMIT BACnet-SimpleACK-PDU
5. IF (Out_Of_Service is writable) THEN
   WRITE X, Out_Of_Service = TRUE
   BEFORE **Notification Fail Time**
   RECEIVE ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = (the same value used in step 1),
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = X,
   'Time Remaining' = (any value appropriate for the Lifetime selected),

'List of Values' = (~~the initial~~*ReportedPV = the current* Present_Value, ~~and new~~ *Current* Status_Flags)
TRANSMIT BACnet-SimpleACK-PDU
6. IF (Present_Value is now writable) THEN
WRITE X, Present_Value = (any value that differs from ~~"initial Present_Value"~~ *ReportedPV*)
ELSE
MAKE (Present_Value = any value that differs from ~~"initial Present_Value"~~ *ReportedPV*)
7. BEFORE **Notification Fail Time**
RECEIVE ConfirmedCOVNotification-Request,
'Subscriber Process Identifier' = (the same value used in step 1),
'Initiating Device Identifier' = IUT,
'Monitored Object Identifier' = X,
'Time Remaining' = (any value appropriate for the Lifetime selected),
'List of Values' = (the new Present_Value and ~~new~~ *Current* Status_Flags)
8. TRANSMIT BACnet-SimpleACK-PDU
9. TRANSMIT SubscribeCOV-Request,
'Subscriber Process Identifier' = (the same value used in step 1),
'Monitored Object Identifier' = X
10. RECEIVE BACnet-SimpleACK-PDU
11. IF (Out_Of_Service is writable) THEN
WRITE X, Out_Of_Service = FALSE
~~RECEIVE BACnet-SimpleACK-PDU~~

8.2.4 Change of Value Notification from a Binary Input, Binary Output, and Binary Value Object Status_Flags Property

Reason for Change: Updated 'Test Concept' to include case if finite lifetime is not supported. Removed extraneous SimpleACKs that appear after WRITE statements.. Updated 'Configuration Requirements'.

Purpose: To verify that the IUT can initiate ConfirmedCOVNotification service requests conveying a change of the Status_Flags property of Binary Input, Binary Output, and Binary Value objects.

Test Concept: A subscription for COV notifications is established, using a Lifetime of L. ~~Removed extraneous SimpleACKs after WRITE statements.~~ *L shall be set to a value less than 24 hours and large enough to complete the test.* The Status_Flags property of the monitored object is then changed and a notification shall be received. The value of the Status-_Flags property can be changed by using the WriteProperty service or by another means. For some implementations writing to the Out_Of_Service property will accomplish this task. For implementations where it is not possible to write to Status_Flags or Out_Of_Service or change the Status_Flags by any other means, this test shall be skipped.

Configuration Requirements: At the beginning of the test, the Out_Of_Service property shall have a value of FALSE. *Select an object where Present_Value is not expected to change outside the tester's control* ~~*by more than COV Increment*~~ *or which has a writable Out_Of_Service.*

Test Steps:

REPEAT X = (one supported object of each type from the set Binary Input, Binary Output, and Binary Value) DO {

1. TRANSMIT SubscribeCOV-Request,
'Subscriber Process Identifier' = (any value > 0 chosen by the TD),
'Monitored Object Identifier' = X,
'Issue Confirmed Notifications' = TRUE,
'Lifetime' = L
2. RECEIVE BACnet-SimpleACK-PDU
3. BEFORE **Notification Fail Time**
RECEIVE ConfirmedCOVNotification-Request,
'Subscriber Process Identifier' = (the same value used in step 1),
'Initiating Device Identifier' = IUT,
'Monitored Object Identifier' = X,
'Time Remaining' = (any value appropriate for the Lifetime selected),
'List of Values' = (the initial Present_Value and initial Status_Flags)

4. TRANSMIT BACnet-SimpleACK-PDU
5. WRITE X, Out_Of_Service = TRUE | WRITE X, Status_Flags = (a value that differs from initial Status_Flags) | MAKE (Status_Flags = any value that differs from initial Status_Flags)

~~1. IF (WriteProperty is used in step 5) THEN~~
~~RECEIVE BACnet-SimpleACK-PDU~~

~~7~~6. BEFORE **Notification Fail Time**
RECEIVE ConfirmedCOVNotification-Request,
'Subscriber Process Identifier' = (the same value used in step 1),
'Initiating Device Identifier' = IUT,
'Monitored Object Identifier' = X,
'Time Remaining' = (any value appropriate for the Lifetime selected),
'List of Values' = (~~the initial~~*the current* Present_Value, and ~~new~~ *Current* Status_Flags)

~~8~~7. TRANSMIT BACnet-SimpleACK-PDU

~~9~~8. TRANSMIT SubscribeCOV-Request,
'Subscriber Process Identifier' = (the same value used in step 1),
'Monitored Object Identifier' = X

~~10~~9.RECEIVE BACnet-SimpleACK-PDU

~~11~~*10* IF (Out_Of_Service was changed in step 5) THEN
WRITE X, Out_Of_Service = FALSE
~~RECEIVE BACnet-SimpleACK-PDU~~

8.2.5 Change of Value Notification from a Multi-state Input, Multi-state Output, Multi-state Value, Life Safety Point, ~~or~~ Life Safety Zone, *CharacterString Value*, *OctetString Value, Date Value, Date Pattern Value, DateTime Value, DateTime Pattern Value, Time Value, or Time Pattern Value* Object Present_Value Property

Reason for Change: Added more primitive value objects. Updated text for 'List of Values'. Updated 'Configuration Requirements'. Removed extraneous SimpleACKs after WRITE statements.

Purpose: To verify that the IUT can initiate ConfirmedCOVNotification service requests conveying a change of the Present_Value property of Multi-state Input, Multi-state Output, Multi-state Value, Life Safety Point, ~~and~~ Life Safety Zone, *CharacterString Value, OctetString Value, Date Value, Date Pattern Value, DateTime Value, DateTime Pattern Value, Time Value, or Time Pattern Value* objects.

Test Concept: A subscription for COV notifications is established, using a Lifetime of L. L shall be set to a value less than 24 hours and large enough to complete the test. The Present_Value of the monitored object is changed and a notification shall be received. The Present_Value may be changed using the WriteProperty service or by another means such as changing the input signal represented by the object. For some implementations it may be necessary to write to the Out_Of_Service property first to accomplish this task. For implementations where it is not possible to write to these properties at all, the vendor shall provide an alternative trigger mechanism to accomplish this task. All of these methods are equally acceptable.

Configuration Requirements: At the beginning of the test, the Out_Of_Service property shall have a value of FALSE. *Select an object where Present_Value is not expected to change outside the tester's control ~~by more than COV_Increment~~ or which has a writable Out_Of_Service.*

Test Steps:

REPEAT X = (one supported object of each type from the set Multi-state Input, Multi-state Output, Multi-state Value, Life Safety Point, ~~and~~ Life Safety Zone, *CharacterString Value, OctetString Value, Date Value, Date Pattern Value, DateTime Value, DateTime Pattern Value, Time Value, or Time Pattern Value*) DO {

1. TRANSMIT SubscribeCOV-Request,
'Subscriber Process Identifier' = (any value > 0 chosen by the TD),
'Monitored Object Identifier' = X,
'Issue Confirmed Notifications' = TRUE,
'Lifetime' = L
2. RECEIVE BACnet-SimpleACK-PDU
3. BEFORE **Notification Fail Time**
RECEIVE ConfirmedCOVNotification-Request,
'Subscriber Process Identifier' = (the same value used in step 1),

'Initiating Device Identifier' = IUT,
'Monitored Object Identifier' = X,
'Time Remaining' = (any value appropriate for the Lifetime selected),
'List of Values' = (the initial Present_Value and initial Status_Flags)

4. TRANSMIT BACnet-SimpleACK-PDU
5. IF (Out_Of_Service is writable) THEN
WRITE X, Out_Of_Service = TRUE
~~RECEIVE BACSimpleACK-PDU~~
BEFORE **Notification Fail Time**
RECEIVE ConfirmedCOVNotification-Request,
'Subscriber Process Identifier' = (the same value used in step 1),
'Initiating Device Identifier' = IUT,
'Monitored Object Identifier' = X,
'Time Remaining' = (any value appropriate for the Lifetime selected),
'List of Values' = (~~the initial~~*ReportedPV = the current* Present_Value, and the ~~new~~ *Current* Status_Flags)
TRANSMIT BACnet-SimpleACK-PDU
6. IF (Present_Value is now writable) THEN
WRITE X, Present_Value = (any value that differs from ~~"initial value"~~*ReportedPV*)
ELSE
MAKE (Present_Value = any value that differs from ~~"initial value"~~*ReportedPV*)
7. BEFORE **Notification Fail Time**
RECEIVE ConfirmedCOVNotification-Request,
'Subscriber Process Identifier' = (the same value used in step 1),
'Initiating Device Identifier' = IUT,
'Monitored Object Identifier' = X,
'Time Remaining' = (any value appropriate for the Lifetime selected),
'List of Values' = (the new Present_Value and ~~new~~ *Current* Status_Flags)
8. TRANSMIT BACnet-SimpleACK-PDU
9. TRANSMIT SubscribeCOV-Request,
'Subscriber Process Identifier' = (the same value used in step 1),
'Monitored Object Identifier' = X
10. RECEIVE BACnet-SimpleACK-PDU
11. IF (Out_Of_Service is writable) THEN
WRITE X, Out_Of_Service = FALSE
~~RECEIVE BACnet-SimpleACK-PDU~~

8.2.6 Change of Value Notification from a Multi-state Input, Multi-state Output Multi-state Value, Life Safety Point, ~~and~~ Life Safety Zone, *CharacterString Value, OctetString Value, Date Value, Date Pattern Value, DateTime Value, DateTime Pattern Value, Time Value, or Time Pattern Value* Object Status_Flags Property

Reason for Change: Added more primitive value objects. Updated Configuration Requirements. Modified text for 'List of Values' in step 7. Removed extraneous SimpleACKs after WRITE statements.

Purpose: To verify that the IUT can initiate ConfirmedCOVNotification service requests conveying a change of the Status_Flags property of Multi-state Input, Multi-state Output, Multi-state Value, Life Safety Point, ~~and~~ Life Safety Zone, *CharacterString Value, OctetString Value, Date Value, Date Pattern Value, DateTime Value, DateTime Pattern Value, Time Value, or Time Pattern Value* objects.

Test Concept: A subscription for COV notifications is established, using a Lifetime of L. L shall be set to a value less than 24 hours and large enough to complete the test. The Status_Flags property of the monitored object is then changed and a notification shall be received. The value of the Status_Flags property can be changed by using the WriteProperty service or by another means. For some implementations writing to the Out_Of_Service property will accomplish this task. For implementations where it is not possible to write to Status_Flags or Out_Of_Service or change the Status_Flags by any other means, this test shall be skipped.

Configuration Requirements: At the beginning of the test, the Out_Of_Service property shall have a value of FALSE. *Select an object where Present_Value is not expected to change outside the tester's control ~~by more than COV_Increment~~ or which has a writable Out_Of_Service.*

Test Steps:

REPEAT X = (one supported object of each type from the set Multi-state input, Multi-state Output, Multi-state Value, Life Safety Point, ~~and~~ Life Safety Zone, *CharacterString Value, OctetString Value, Date Value, Date Pattern Value, DateTime Value, DateTime Pattern Value, Time Value, or Time Pattern Value*) DO {

1. TRANSMIT SubscribeCOV-Request,
   'Subscriber Process Identifier' = (any value > 0 chosen by the TD),
   'Monitored Object Identifier' = X,
   'Issue Confirmed Notifications' = TRUE,
   'Lifetime' = L
2. RECEIVE BACnet-SimpleACK-PDU
3. BEFORE **Notification Fail Time**
   RECEIVE ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = (the same value used in step 1),
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = X,
   'Time Remaining' = (any value appropriate for the Lifetime selected),
   'List of Values' = (the initial Present_Value and initial Status_Flags)
4. TRANSMIT BACnet-SimpleACK-PDU
5. WRITE X, Out_Of_Service = TRUE | WRITE X, Status_Flags = (a value that differs from initial Status_Flags) | MAKE (Status_Flags = any value that differs from initial Status_Flags)
~~2. IF (WriteProperty is used in step 5) THEN~~
~~RECEIVE BACnet-SimpleACK-PDU~~
7. BEFORE **Notification Fail Time**
   RECEIVE ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = (the same value used in step 1),
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = X,
   'Time Remaining' = (any value appropriate for the Lifetime selected),
   'List of Values' = (~~the initial~~*the current* Present_Value, and ~~new~~ *Current* Status_Flags)
8. TRANSMIT BACnet-SimpleACK-PDU
9. TRANSMIT SubscribeCOV-Request,
   'Subscriber Process Identifier' = (the same value used in step 1),
   'Monitored Object Identifier' = X
10. RECEIVE BACnet-SimpleACK-PDU
11. IF (Out_Of_Service was changed in step 5) THEN
    WRITE X, Out_Of_Service = FALSE
    ~~RECEIVE BACnet-SimpleACK-PDU~~

BTL-15.1f-6: Allow Devices to Cancel COV

Overview:

Clarification request CR-0421 made it known that tests 9.2.2.1 and 8.11.X1.3 do not make allowances for devices that actively cancel COV subscriptions rather than let them expire. Actively canceling a COV subscription is the preferred practice and, as the testable point only requires that the subscription no longer exists, the tests are modified to make this allowance. Test 9.2.2.1 was also missing a 'Test Concept' so that has been added.

[In BTL Specified Tests, change test 9.2.2.1 and 8.11.X1.3 as shown]

9.2.2.1 Change of Value Notification Arrives after Subscription has Expired

Reason for Change: Corrected tests per BTL-CR-0299 and added Configuration Requirements section.

Purpose: To verify that an appropriate error is returned if a COV notification arrives after the subscription time period has expired.

Test Concept: A subscription for COV notifications is established and then cancelled or allowed to expire. A ConfirmedCOVNotification is then sent to the IUT to verify it returns the appropriate error or a Simple-Ack.

Configuration Requirements: If the IUT does not support initiation of SubscribeCOV-Request with 'Issue Confirmed Notifications' equal to TRUE, then this test shall be skipped.

Test Steps:

1. RECEIVE SubscribeCOV-*Request*,
   'Subscriber Process Identifier' = (any valid process identifier*, PI*),
   'Monitored Object Identifier' = (any object X of a type that supports COV notification),
   'Issue Confirmed Notifications ' = TRUE,
   'Lifetime' = (~~a value no greater than one minute~~*any valid Lifetime*)
2. TRANSMIT BACnet-SimpleACK-PDU
3. *TRANSMIT ConfirmedCOVNotification-Request*,
   'Subscriber Process Identifier' = (*~~the process identifier used in step 1~~PI*),
   'Initiating Device Identifier' = TD,
   'Monitored Object Identifier' = X,
   'Time Remaining' = (any amount of time greater than 0),
   'List of Values' = (a list of values appropriate to object X)

4. IF (the IUT can cancel the subscription)
   RECEIVE SubscribeCOV – Request,
   'Subscriber Process Identifier' = (PI),
   'Monitored Object Identifier' = X
   ELSE
   MAKE (the IUT stop resubscribing, if it resubscribes automatically)

~~5~~3. WAIT (~~a value two times~~ *at least* Lifetime, *but sufficient to ensure the subscription has expired*)

~~6~~4. TRANSMIT ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = (~~the process identifier used in step 2~~*1PI*),
   'Initiating Device Identifier' = TD,
   'Monitored Object Identifier' = X,
   'Time Remaining' = (any amount of time greater than 0),
   'List of Values' = (a list of values appropriate to object X)

~~7~~5. IF (Protocol_Revision is present and Protocol_Revision >= 10) THEN
   RECEIVE BACnet-Error-PDU,
   Error Class = SERVICES,
   Error Code = UNKNOWN_SUBSCRIPTION |
   (BACnet-SimpleACK-PDU)
   ELSE
   RECEIVE BACnet-Error-PDU,
   Error Class = SERVICES,
   Error Code = (any valid error code for class SERVICES) |

(BACnet-SimpleACK-PDU)

[In BTL Specified Tests, add this test.]

8.11.X1.3 Change of Value Notification Arrives after Subscription has Expired

Reason for Change: Added new test to support DS-COVP-A testing.

Purpose: To verify that an appropriate error is returned if a COV notification arrives after the subscription time period has expired.

Test Concept: A subscription for COV notifications is established and then cancelled or allowed to expire.~~, using a Lifetime of L. L shall be set to a value less than 24 hours and large enough to complete the test.~~ A ConfirmedCOVNotification is then sent to the IUT to verify it returns the appropriate error or a Simple-Ack.

Test Steps:

1. MAKE (the IUT send a SubscribeCOVProperty-Request),
2. RECEIVE SubscribeCOVProperty-Request
   'Subscriber Process Identifier' = (any valid process identifier),
   'Monitored Object Identifier' = X
   'Issue Confirmed Notifications' = TRUE,
   'Lifetime' = L,
   'Monitored Property Identifier' = (the property Y to be monitored),
   'COV Increment' = (Any REAL value -- optional)
3. TRANSMIT BACnet-SimpleACK-PDU
4. BEFORE Notification Fail Time
   TRANSMIT ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = (the process identifier used in step 1),
   'Initiating Device Identifier' = TD,
   'Monitored Object Identifier' = X
   'Time Remaining' = (any value appropriate for the Lifetime selected),
   'List of Values' = (values appropriate to the property Y subscribed to, and any other properties the IUT provides with it, such as Status_Flags)
   RECEIVE BACnet-SimpleACK-PDU
5. IF (the IUT can cancel the subscription)
   RECEIVE SubscribeCOVProperty – Request,
   'Subscriber Process Identifier' = (the process identifier used in step 1),
   'Monitored Object Identifier' = X
   'Monitored Property Identifier' = Y
   'COV Increment' = (Any REAL value –optional)
   ELSE
   WAIT (a value two times Lifetime)
6. TRANSMIT ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = (the process identifier used in step 1),
   'Initiating Device Identifier' = TD,
   'Monitored Object Identifier' = X
   'Time Remaining' = (any value appropriate for the Lifetime selected),
   'List of Values' = (values appropriate to the property Y subscribed to, and any other properties the IUT provides with it, such as Status_Flags)
7. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
   RECEIVE
   BACnet-Error-PDU,
   Error Class = SERVICES,
   Error Code = (UNKNOWN_SUBSCRIPTION) |
   (BACnet-SimpleACK-PDU)
   ELSE
   RECEIVE BACnet-Error-PDU,
   Error Class = SERVICES,
   Error Code = (any valid error code for class SERVICES) |
   (BACnet-SimpleACK-PDU)

BTL-15.1f-7: ReadRange Support for ALL List properties

Overview:

Test 9.21.1.X1 does not specify the expected response when the list being read is empty.

[In BTL Specified Tests, change test 9.21.1.X1 as shown]

9.21.1.X1 ReadRange Support for All List Properties

Reason for change: Need a ReadRange test for non-Log_Buffer list properties.

Purpose: To verify that all list properties of all objects can be read using the 3 by position forms of the ReadRange service.

Configuration Requirements: The IUT must be configured with at least one non-empty list property.

Test Steps:

1. REPEAT X = (all objects in the IUT's database) DO {
   REPEAT Y = (all list properties in object X) DO {
   TRANSMIT ReadRange-Request
   'Object Identifier' = X,
   'Property Identifier' = Y,
   RECEIVE (ReadRange-ACK
   'Object Identifier' = X,
   'Property Identifier' = Y,
   'Result Flags' = (?, ?, ?),
   'Item Count' = (C: up to number of items in Y)
   'Item Data' = (the first C elements of Y)) |
   (ReadRange-ACK
   'Object Identifier' = X,
   'Property Identifier' = Y,
   ('Result Flags' = (FALSE, FALSE, FALSE),
   'Item Count' = (C = 0)
   'Item Data' = ())
   IF (C <> 0) THEN
   TRANSMIT ReadRange-Request
   'Object Identifier' = X,
   'Property Identifier' = Y,
   'Reference Index' = 1,
   'Count' = (C: any valid positive value)
   RECEIVE ReadRange-ACK
   'Object Identifier' = X,
   'Property Identifier' = Y,
   'Result Flags' = (TRUE, ?, ?),
   'Item Count' = (C2: up to C)
   'Item Data' = (the first C2 elements of Y)
   TRANSMIT ReadRange-Request
   'Object Identifier' = X,
   'Property Identifier' = Y,
   'Reference Index' = (the number of elements in Y),
   'Count' = (C: any valid negative value)
   RECEIVE ReadRange-ACK
   'Object Identifier' = X,
   'Property Identifier' = Y,
   'Result Flags' = (?, TRUE, ?),
   'Item Count' = (C2: up to abs(C))
   'Item Data' = (the last C2 elements of Y)
   }
   }