BACnet® TESTING LABORATORIES ADDENDA

Addendum p to BTL Test Package 15

Revision 7
Revised 8/7/2019

Approved by the BTL Working Group on June 20, 2019
Approved by the BTL Working Group Voting Members on August 6, 2019;
Published on August 15, 2019.

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

FOREWORD

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

In the following document, language to be added to existing clauses within the BTL Test Package 15.2 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-15.2p-1: Add Array Test Cases [BTLWG-182]

Overview:
Added new Array tests.

Changes:
[In Test Plan sections 4.2.1 add a test case as shown and remove 3 test cases shown in strikethrough]

4.2 Data Sharing - ReadProperty - B

4.2.1 Base Requirements
All devices must support this BIBB.

BTL - 7.1.1 - Read Support Test Procedure		
	Test Conditionality	Must be executed. To satisfy this test item, this test needs only be executed using ReadProperty.
	Test Directives	
	Testing Hints	.
~~**135.1-2013 - 9.18.1.1 - Reading the Size of an Array**~~		
	~~**Test Conditionality**~~	~~If the test reads for 7.1 reads array sizes, this test can be skipped.~~
	~~**Test Directives**~~	
	~~**Testing Hints**~~	
~~**BTL - 9.18.1.2 - Reading a Single Element of an Array**~~		
	~~**Test Conditionality**~~	~~If the test reads for 7.1 reads individual array elements, this test can be skipped.~~
	~~**Test Directives**~~	
	~~**Testing Hints**~~	
135.1-2013 - 9.18.2.1 - Reading Non-Array Properties with an Array Index		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
~~**135.1-2013 - 9.18.2.2 - Reading Array Properties with an Array Index that is Out of Range**~~		
	~~**Test Conditionality**~~	~~Must be executed.~~
	~~**Test Directives**~~	
	~~**Testing Hints**~~	
135.1-2013 - 9.18.2.3 - Reading an Unknown Object		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
135.1-2013 - 9.18.2.4 - Reading an Unknown Property		
	Test Conditionality	Must be executed.
	Test Directives	Be sure to test at least one property identifier that is within the ASHRAE allocated range for standard property identifiers, but that has not yet been defined.
	Testing Hints	
135.1-2013 - 9.18.1.3 - Reading a Property From the Device Object using the Unknown Instance		
	Test Conditionality	If the device implements protocol revision 4 or higher, this test must be executed.
	Test Directives	
	Testing Hints	
BTL - 7.1.X3 - Verifying Property_List against the EPICS		
	Test Conditionality	Must be executed if the IUT claims Protocol_Revision 14 or greater.
	Test Directives	
	Testing Hints	
BTL - 9.18.1.X4 - Reading Array Properties at different Array Indexes		
	Test Conditionality	*Must be executed.*
	Test Directives	*Repeat for all supported BACnetARRAY properties*

	Testing Hints	

[In BTL Specified Tests in section 9.18.1, add the new testcase as shown]

9.18.1.X4 Reading Array Properties at different Array Indexes

Purpose: This test verifies the IUT can execute ReadProperty service requests on a single element in an array property.

Test Concept: This test will execute a ReadProperty service request to read a single element from the selected property by specifying the array-index in the request.. Another request is made to read an element of an array where the array index is out of range.
Configuration Requirement: O1 is any object in the IUT database having array property P1 having size X.

Test Steps:

1. VERIFY P1 = X, ARRAY INDEX = 0
2. IF (X>0) THEN
   READ V = P1, ARRAY_INDEX = 1
   CHECK (V is any valid value of the correct data type for property P1)
   READ V = P1, Array Index =X
   CHECK (V is any valid value of the correct data type for property P1)
3. TRANSMIT ReadProperty-Request,
   'Object Identifier' = O1,
   'Property Identifier' = P1
   'Array Index' = (X+1)
4. RECEIVE BACnet-Error-PDU,
   Error Class = PROPERTY,
   Error Code = INVALID_ARRAY_INDEX

BTL-15.2p-2: Backup and Restore Test Corrections [BTLWG-445]

Overview:
This document fixes the Backup and Restore tests where the device can respond within APDU_Timeout and optionality of Backup_And_Restore_State prior to rev 13.

Changes:

[In BTL Test Plan, update the test references shown below from 135.1-2013 to BTL]

8.18 Device Management - Backup and Restore - B

8.18.1 Base Requirements

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

BTL - 13.8.1.1 - Execution of Full Backup and Restore Procedure		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
~~135.1-2013~~*BTL* - 13.8.1.2 - Attempting Backup While Already Performing a Backup Procedure		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
~~135.1-2013~~*BTL* - 13.8.1.3 - Attempting Backup While Already Performing a Restore Procedure		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
~~135.1-2013~~*BTL* - 13.8.1.4 - Attempting Restore While Already Performing a Backup Procedure		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
~~135.1-2013~~*BTL* - 13.8.1.5 - Attempting Restore While Already Performing a Restore Procedure		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
BTL - 13.8.1.6 - Ending Backup and Restore Procedures via Timeout		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
~~135.1-2013~~*BTL* - 13.8.1.7 - Ending Backup and Restore Procedures via Abort		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

8.18.3 Supports Non-Password Protected Backup

The IUT does not require, or can be made to not require, a password for a ReinitializeDevice <STARTBACKUP> service request.

~~135.1-2013~~*BTL* - 13.8.1.10 - Executing and Ending a Backup Procedure when a password is not required		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

8.18.6 Changes Operational Behavior during a Backup Procedure

The IUT changes its operational behavior during a Backup Procedure.

~~135.1-2013~~*BTL* - 13.8.1.12 - System_Status during a Backup Procedure		
	Test Conditionality	Must be executed.
	Test Directives	

	Testing Hints	

8.18.7 Changes Operational Behavior during a Restore Procedure

The IUT changes its operational behavior during a Restore Procedure.

~~135.1-2013~~*BTL* - 13.8.1.13 - System_Status during a Restore Procedure		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

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

13.8.1.1 Execution of Full Backup and Restore Procedure

Reason for Change: Corrected the Backup_And_Restore_State in step 22. *Changed test to account for optional properties.*

Purpose: This test case verifies that the IUT can execute a full Backup and Restore procedure.

Test Concept: This test takes the IUT through a successful Backup and then a successful Restore procedure. The Database_Revision and Last_Restore_Time properties are noted before the procedure begins for later comparison. The IUT is then commanded to enter the Backup state; all the files are read, and the IUT is commanded to end the backup. If the Database_Revision property can be changed by means other than the restore procedure, it is modified and checked to ensure that it incremented correctly; then the IUT is commanded to enter the Restore state. If the file objects do not exist on the IUT, the TD will create them in the IUT. The files are then truncated to size 0, the file contents are written to the IUT, and the IUT is commanded to end the restore. The Database_Revision and Last_Restore_Time properties are checked to ensure that they incremented or advanced correctly.

For IUTs that use Stream Access when performing the AtomicReadFile and AtomicWriteFile services, a Maximum Requested Octet Count (MROC) and a Maximum Write Data Length (MWDL) shall be calculated before starting the test. These values shall be used during the test. MROC shall be 16 less than the minimum of the TD's Max_APDU_Length_Accepted and the IUT's maximum transmittable APDU length. MWDL shall be 21 less than the minimum of the TD's maximum transmittable APDU length and the IUT's Max_APDU_Length_Accepted.

Test Steps:

1. READ DR1 = Database_Revision
2. READ LRT1 = Last_Restore_Time
3. READ OL1 = Object_List
4. REPEAT X = (1 through length of OL1) DO {
   READ NAMES[X] = (OL1[X]), Object_Name
   }
5. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
   IF (Backup_Preparation_Time is present) THEN
   READ BPT = Backup_Preparation_Time
   ELSE
   READ BPT = APDU_Timeout
   IF (Restore_Preparation_Time is present) THEN
   READ RPT = Restore_Preparation_Time
   ELSE
   READ RPT = APDU_Timeout
   IF (Restore_Completion_Time is present) THEN
   READ RCT = Restore_Completion_Time
   ELSE
   READ RCT = APDU_Timeout
   IF (Backup_And_Restore_State is present or Protocol_Revision ≥ 13) THEN
   VERIFY Backup_And_Restore_State = IDLE
6. TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = STARTBACKUP,
   'Password' = (any valid password)

7. RECEIVE BACnet-Simple-ACK-PDU
8. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
    WAIT BPT
    IF (Backup_And_Restore_State is present or Protocol_Revision ≥ 13) THEN
        READ BRSTATE = Backup_And_Restore_State
        READ CF = Configuration_Files
        WHILE (BRSTATE = PREPARING_FOR_BACKUP) DO {
            WAIT 1 second
            READ BRSTATE = Backup_And_Restore_State
            IF CF is an empty list THEN
                READ CF = Configuration_Files
                IF CF is a non-empty list THEN
                    READ X = (the file referenced by Configuration_Files[1]).Name
        }
        CHECK (BRSTATE = PERFORMING_A_BACKUP)
9. READ CF = Configuration_Files
10. CHECK (CF is a non-empty array of BACnetObjectIdentifiers referring to File objects)
11. REPEAT X = (each entry in CF) DO {
    READ Y = X, File_Access_Method
    IF (Y = RECORD_ACCESS)
        WHILE (the last read resulted in an Ack with 'End Of File' == FALSE) DO {
            TRANSMIT AtomicReadFile-Request,
                'Object Identifier' = X,
                'File Start Record' = (the next unread record),
                'Requested Record Count' = 1
            RECEIVE AtomicReadFile-ACK,
                'End Of File' = TRUE | FALSE,
                'File Start Record' = Z,
                'Requested Record Count' = 1
                'Returned Data' = (File contents)
            | Error-PDU -- only acceptable for the first record and only when there are no records in the file
                'Error Class' = SERVICES,
                'Error Code' = INVALID_FILE_START_POSITION
        }
    ELSE
        WHILE (the last read did not indicate 'End Of File') DO {
            TRANSMIT AtomicReadFile-Request,
                'Object Identifier' = X,
                'File Start Position' = (the next unread octet),
                'Requested Octet Count' = MROC
            RECEIVE AtomicReadFile-ACK,
                'End Of File' = TRUE | FALSE,
                'File Start Position' = (the next unread octet)
                'File Data' = (File contents of length MROC if 'End Of File' is FALSE
                    or of length MROC or less if 'End Of File' is TRUE)
            | Error-PDU -- only acceptable for the first record and only when there are no records in the file
                'Error Class' = SERVICES,
                'Error Code' = INVALID_FILE_START_POSITION
        }
    }
12. TRANSMIT ReinitializeDevice-Request,
    'Reinitialize State Of Device' = ENDBACKUP,
    'Password' = (any valid password)
13. RECEIVE BACnet-Simple-ACK-PDU
14. VERIFY System_Status ! = BACKUP_IN_PROGRESS
15. IF (*Backup_And_Restore_State is present or (*Protocol_Revision is present and Protocol_Revision ≥ ~~10~~*13)*) THEN
    VERIFY Backup_And_Restore_State = IDLE
16. IF (Database_Revision is changeable) THEN
    MAKE (the configuration in the IUT different, such that the Database_Revision property increments)
    VERIFY Database_Revision <> DR1

READ DR2 = Database_Revision
CHECK (DR1 <> DR2)

17. TRANSMIT ReinitializeDevice-Request,
'Reinitialize State Of Device' = STARTRESTORE,
'Password' = (any valid password)
18. RECEIVE BACnet-Simple-ACK-PDU
19. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
WAIT RPT
IF (Backup_And_Restore_State is present or Protocol_Revision ≥ 13) THEN
READ BRSTATE = Backup_And_Restore_State
WHILE (BRSTATE = PREPARING_FOR_RESTORE) DO {
WAIT 1 second
READ BRSTATE = Backup_And_Restore_State
}
CHECK (BRSTATE = PERFORMING_A_RESTORE)
20. READ OL2 = Object_List
21. REPEAT X = (entry in CF) DO {
IF (X is not in OL2)
TRANSMIT CreateObject-Request
'Object Identifier' = X
RECEIVE CreateObject-ACK
'Object Identifier' = X
READ FS = X, File_Size
IF (File_Size is not equal to the size of the backed up file)
WRITE X, File_Size = 0
IF (Y = RECORD_ACCESS)
TRANSMIT AtomicWriteFile-Request
'File Identifier' = X
'File Start Record' = 0
'Record Data' = (file content for first record obtained in step 11)
RECEIVE AtomicWriteFile-ACK
'File Start Record' = 0
REPEAT REC = (each record in the backup of this file) {
TRANSMIT AtomicWriteFile-Request
'File Identifier' = X
'File Start Record' = -1
'Record Count' = 1
'Record Data' = REC
RECEIVE AtomicWriteFile-ACK
'File Start Record' = (the record number)
}
ELSE
REPEAT Z = (0 through the file size, in increments of MWDL) DO {
TRANSMIT AtomicWriteFile-Request
'File Identifier' = X
'File Start Position' = Z
'Record Data' = (file contents obtained from the backup, the number of octets
being the lesser of (file size - Z) and MWDL)
RECEIVE AtomicWriteFile-ACK
'File Start Position' = Z
}
}
22. IF (*Backup_And_Restore_State is present or (*Protocol_Revision is present and Protocol_Revision ≥ ~~10~~*13)*) THEN
VERIFY Backup_And_Restore_State = ~~RESTORE_IN_PROGRESS~~ *PERFORMING_A_RESTORE*
23. TRANSMIT ReinitializeDevice-Request,
'Reinitialize State Of Device' = ENDRESTORE,
'Password' = (any valid password)
24. RECEIVE BACnet-Simple-ACK-PDU
25. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
WAIT RCT

IF (Backup_And_Restore_State is present or Protocol_Revision ≥ 13) THEN
VERIFY Backup_And_Restore_State = IDLE
26. READ DR3 = Database_Revision
27. CHECK (DR3 <> DR1)
28. IF (Database_Revision was changed in step 16) THEN
CHECK (DR3 <> DR2)
29. VERIFY Last_Restore_Time > LRT1
30. READ OL3 = Object_List
31. CHECK (that OL1 and OL3 contain the same set of objects)
32. REPEAT X = (1 through length of OL1) DO {
VERIFY (OL1[X]), Object_Name = NAMES[X]
}

13.8.1.6 Ending Backup and Restore Procedures via Timeout

Reason for Change: Change: Modified how the test WAITs for Protocol_Revision < 10. *Changed test to account for optional properties.*

Purpose: This test case verifies that the IUT will end Backup and Restore procedures after not receiving any messages related to the backup or restore for longer than Backup_Failure_Timeout and that the Backup_Failure_Timeout property is writeable.

Test Steps:

1. WRITE Backup_Failure_Timeout = (A value T1 greater than Backup_Preparation_Timeout)
2. VERIFY Backup_Failure_Timeout = T1
3. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
IF (Backup_Preparation_Time is present) THEN
READ BPT = Backup_Preparation_Time
ELSE
READ BPT = APDU_Timeout
4. TRANSMIT ReinitializeDevice-Request,
'Reinitialized State of Device' = STARTBACKUP,
'Property Identifier' = (any valid password)
5. RECEIVE Simple-ACK-PDU
6. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
WAIT BPT
IF (Backup_And_Restore_State is present or Protocol_Revision ≥ 13) THEN
READ BRSTATE = Backup_And_Restore_State
WHILE (BRSTATE = PREPARING_FOR_BACKUP) DO {
WAIT 1 second
READ BRSTATE = Backup_And_Restore_State
}
CHECK (BRSTATE = PERFORMING_A_BACKUP)
7. WAIT (T1 + 10 seconds)
8. IF (*Backup_And_Restore_State is present or (*Protocol_Revision is present and Protocol_Revision ≥ ~~*10*~~*13)*) THEN
VERIFY Backup_And_Restore_State = IDLE
9. VERIFY System_Status ! = BACKUP_IN_PROGRESS
10. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
IF (Restore_Preparation_Time is present) THEN
READ RPT = Restore_Preparation_Time
ELSE
READ RPT = APDU_Timeout
IF (Restore_Completion_Time is present) THEN
READ RCT = Restore_Completion_Time
ELSE
READ RCT = APDU_Timeout
11. TRANSMIT ReinitializeDevice-Request,
'Reinitialized State of Device' = STARTRESTORE,
'Password' = (any valid password)
12. RECEIVE BACnet-Simple ACK-PDU

13. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
    WAIT RPT
    IF (Backup_And_Restore_State is present or Protocol_Revision ≥ 13) THEN
        READ BRSTATE = Backup_And_Restore_State
        WHILE (BRSTATE = PREPARING_FOR_RESTORE) DO {
          WAIT 1 second
          READ BRSTATE = Backup_And_Restore_State
        }
        CHECK (BRSTATE = PERFORMING_A_RESTORE)
    ELSE
        WAIT (30 seconds)
14. WAIT (*T1 + 10* ~~40~~ seconds)
15. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
    WAIT RCT
    IF (Backup_And_Restore_State is present or Protocol_Revision ≥ 13) THEN
        VERIFY Backup_And_Restore_State = IDLE
16. VERIFY System_Status ! = DOWNLOAD_IN_PROGRESS

Notes to Tester: After an incomplete restore attempt, the IUT may revert to a default configuration or another state that is different from the IUT state when this test was started.

13.8.1.11 Starting and Ending a Restore Procedure when a Password is not Required

Reason For Change: Corrected the 'Reinitialized State of Device' value in step 5. *Changed test to account for optional properties.*

Purpose: This test case verifies that the IUT ignores the password. If the IUT cannot be made to accept a ReinitializeDevice service request that contains any or no password, then this test shall be omitted.

Test Steps:

1. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
   IF (Restore_Preparation_Time is present) THEN
       READ RPT = Restore_Preparation_Time
   ELSE
       READ RPT = APDU_Timeout
   IF (Restore_Completion_Time is present) THEN
       READ RCT = Restore_Completion_Time
   ELSE
       READ RCT = APDU_Timeout
2. TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = STARTRESTORE,
   'Password' = (any non-zero length password)
3. RECEIVE BACnet-Simple ACK-PDU
4. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
       WAIT RPT
5. TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = ~~END~~*ABORT*RESTORE,
   'Password' = (any non-zero length password)
6. RECEIVE BACnet-Simple ACK-PDU
7. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
       WAIT RCT
8. VERIFY System_Status ! = DOWNLOAD_IN_PROGRESS

[In BTL Specified Tests, add the below tests]

13.8.1.2 Attempting a Backup Procedure While Already Performing a Backup Procedure

Reason for Change: Changed test to account for optional properties.

Purpose: To verify that the IUT correctly rejects a command to start a Backup Procedure when already performing a Backup Procedure.

Test Concept: The IUT is commanded to start a Backup Procedure from one client and then is commanded to start a Backup Procedure from a different client.

Test Steps:

1. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
   IF (Backup_Preparation_Time is present) THEN
   READ BPT = Backup_Preparation_Time
   ELSE
   READ BPT = APDU_Timeout
2. TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = STARTBACKUP,
   'Property Identifier' = (any valid password)
3. RECEIVE Simple-ACK-PDU
4. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
   WAIT BPT
5. TRANSMIT
   SNET = (N, any remote network number),
   SADR = (M, any MAC address valid for the specified network),
   ReinitializeDevice-Request,
   'Reinitialized State of Device' = STARTBACKUP,
   'Property Identifier' = (any valid password),
6. RECEIVE
   DNET = N,
   DADR = M,
   BACnet-Error PDU,
   Error Class = DEVICE,
   Error Code = CONFIGURATION_IN_PROGRESS
7. TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = ENDBACKUP,
   'Property Identifier' = (any valid password)
8. RECEIVE Simple-ACK-PDU

13.8.1.3 Attempting a Backup Procedure While Already Performing a Restore Procedure

Reason for Change: Changed test to account for optional properties.

Purpose: To verify that the IUT correctly rejects a command to start a Backup Procedure when already performing a Restore Procedure.

Test Steps:

1. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
   IF (Restore_Preparation_Time is present) THEN
   READ RPT = Restore_Preparation_Time
   ELSE
   READ RPT = APDU_Timeout
   IF (Restore_Completion_Time is present) THEN
   READ RCT = Restore_Completion_Time
   ELSE
   READ RCT = APDU_Timeout
2. TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = STARTRESTORE,
   'Property Identifier' = (any valid password)
3. RECEIVE Simple-ACK-PDU

4. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
   WAIT RPT
5. TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = STARTBACKUP,
   'Property Identifier' = (any valid password),
6. RECEIVE BACnet-Error PDU,
   Error Class = DEVICE,
   Error Code = CONFIGURATION_IN_PROGRESS
7. TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = ABORTRESTORE,
   'Property Identifier' = (any valid password)
8. RECEIVE Simple-ACK-PDU
9. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
   WAIT RCT

Note to Tester: After an incomplete restore attempt, the IUT may revert to a default configuration or another state that is different from the IUT state when this test was started.

13.8.1.4 Attempting a Restore Procedure While Already Performing a Backup Procedure

Reason for Change: Changed test to account for optional properties.
Purpose: To verify that the IUT correctly rejects a command to start a Restore Procedure when already performing a Backup Procedure.

Test Concept: The IUT is commanded to start a Restore Procedure from one client and then is commanded to start a Restore Procedure from a different client.

Test Steps:

1. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
   IF (Backup_Preparation_Time is present) THEN
      READ BPT = Backup_Preparation_Time
   ELSE
      READ BPT = APDU_Timeout
2. TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = STARTBACKUP,
   'Property Identifier' = (any valid password)
3. RECEIVE Simple-ACK-PDU
4. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
   WAIT BPT
5. TRANSMIT
   SNET = (N, any remote network number),
   SADR = (M, any MAC address valid for the specified network),
   ReinitializeDevice-Request,
   'Reinitialized State of Device' = STARTRESTORE,
   'Property Identifier' = (any valid password),
6. RECEIVE
   DNET = N,
   DADR = M,
   BACnet-Error PDU,
   Error Class = DEVICE,
   Error Code = CONFIGURATION_IN_PROGRESS
7. TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = ENDBACKUP,
   'Property Identifier' = (any valid password)
8. RECEIVE Simple-ACK-PDU

13.8.1.5 Attempting a Restore Procedure While Already Performing a Restore Procedure

Reason for Change: Changed test to account for optional properties.

Purpose: To verify that the IUT correctly rejects a command to start a Restore Procedure when already performing a Restore Procedure.

Test Steps:

1. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
   IF (Restore_Preparation_Time is present) THEN
   READ RPT = Restore_Preparation_Time
   ELSE
   READ RPT = APDU_Timeout
   IF (Restore_Completion_Time is present) THEN
   READ RCT = Restore_Completion_Time
   ELSE
   READ RCT = APDU_Timeout
2. TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = STARTRESTORE,
   'Property Identifier' = (any valid password)
3. RECEIVE Simple-ACK-PDU
4. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
   WAIT RPT
5. TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = STARTRESTORE,
   'Property Identifier' = (any valid password),
6. RECEIVE BACnet-Error PDU,
   Error Class = DEVICE,
   Error Code = CONFIGURATION_IN_PROGRESS
7. TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = ABORTRESTORE,
   'Property Identifier' = (any valid password)
8. RECEIVE Simple-ACK-PDU
9. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
   WAIT RCT

Notes to Tester: After an incomplete restore attempt, the IUT may revert to a default configuration or another state that is different from the IUT state when this test was started.

13.8.1.7 Ending Backup and Restore Procedures via Abort

Reason for Change: Changed test to account for optional properties.
Purpose: This test case verifies that the IUT will leave the BACKUP_IN_PROGRESS and DOWNLOAD_IN_PROGRESS states upon a command to abort.

Test Steps:

1. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
   IF (Backup_Preparation_Time is present) THEN
   READ BPT = Backup_Preparation_Time
   ELSE
   READ BPT = APDU_Timeout
2. TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = STARTBACKUP,
   'Password' = (any valid password)
3. RECEIVE BACnet-Simple ACK-PDU
4. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
   WAIT BPT
5. TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = ENDBACKUP,
   'Password' = (any valid password)

6. RECEIVE BACnet-Simple ACK-PDU
7. IF (*Backup_And_Restore_State is present or (*Protocol_Revision is present and Protocol_Revision ≥ ~~10~~*13)*) THEN
   VERIFY Backup_And_Restore_State = IDLE
8. VERIFY System_Status != BACKUP_IN_PROGRESS
9. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
   IF (Restore_Preparation_Time is present) THEN
   READ RPT = Restore_Preparation_Time
   ELSE
   READ RPT = APDU_Timeout
   IF (Restore_Completion_Time is present) THEN
   READ RCT = Restore_Completion_Time
   ELSE
   READ RCT = APDU_Timeout
10. TRANSMIT ReinitializeDevice-Request,
    'Reinitialized State of Device' = STARTRESTORE,
    'Password' = (any valid password)
11. RECEIVE BACnet-Simple ACK-PDU
12. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
    WAIT RPT
13. TRANSMIT ReinitializeDevice-Request,
    'Reinitialized State of Device' = ABORTRESTORE,
    'Password' = (any valid password)
14. RECEIVE BACnet-Simple ACK-PDU
15. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
    WAIT RCT
    IF (Backup_And_Restore_State is present or Protocol_Revision ≥ 13) THEN
    VERIFY Backup_And_Restore_State = IDLE
16. VERIFY System_Status ! = DOWNLOAD_IN_PROGRESS

Notes to Tester: After an incomplete restore attempt, the IUT may revert to a default configuration or another state that is different from the IUT state when this test was started.

13.8.1.10 Starting and Ending a Backup Procedure when a Password is not Required

Reason for Change: Changed test to account for optional properties.

Purpose: This test case verifies that the IUT ignores the password. If the IUT cannot be made to accept a ReinitializeDevice service request that contains any or no password, then this test shall be omitted.

Test Steps:

1. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
   IF (Backup_Preparation_Time is present) THEN
   READ BPT = Backup_Preparation_Time
   ELSE
   READ BPT = APDU_Timeout
2. TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = STARTBACKUP,
   'Password' = (any non-zero length password)
3. RECEIVE BACnet-Simple ACK-PDU
4. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
   WAIT BPT
5. TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = ENDBACKUP,
   'Password' = (any non-zero length password)
6. RECEIVE BACnet-Simple ACK-PDU
7. IF (*Backup_And_Restore_State is present or (*Protocol_Revision is present and Protocol_Revision ≥ ~~10~~*13)*) THEN
   VERIFY Backup_And_Restore_State = IDLE
8. VERIFY System_Status ! = BACKUP_IN_PROGRESS

13.8.1.12 System_Status during a Backup Procedure

Reason for Change: Changed test to account for optional properties.

Purpose: This test case verifies that the IUT correctly sets its System_Status during a Backup procedure. If the IUT does not change its operational behavior during a Backup Procedure, then this test shall be omitted.

Test Steps:

1. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
   IF (Backup_Preparation_Time is present) THEN
   READ BPT = Backup_Preparation_Time
   ELSE
   READ BPT = APDU_Timeout
2. TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = STARTBACKUP,
   'Password' = (any valid password)
3. RECEIVE BACnet-Simple ACK-PDU
4. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
   WAIT BPT
5. VERIFY System_Status = BACKUP_IN_PROGRESS
6. TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = ENDBACKUP,
   'Password' = (any valid password)
7. RECEIVE BACnet-Simple ACK-PDU
8. WAIT a vendor specified period of time for the device to complete the backup operation
9. VERIFY System_Status != BACKUP_IN_PROGRESS

13.8.1.13 System_Status during a Restore Procedure

Reason for Change: Changed test to account for optional properties.

Purpose: This test case verifies that the IUT correctly sets its System_Status during a Restore procedure. If the IUT does not change its operational behavior during a Restore Procedure, this test shall be omitted.

Test Steps:

1. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
   IF (Restore_Preparation_Time is present) THEN
   READ RPT = Restore_Preparation_Time
   ELSE
   READ RPT = APDU_Timeout
   IF (Restore_Completion_Time is present) THEN
   READ RCT = Restore_Completion_Time
   ELSE
   READ RCT = APDU_Timeout
2. TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = STARTRESTORE,
   'Password' = (any valid password)
3. RECEIVE BACnet-Simple ACK-PDU
4. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
   WAIT RPT
5. VERIFY System_Status = DOWNLOAD_IN_PROGRESS
6. TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = ABORTRESTORE,
   'Password' = (any valid password)
7. RECEIVE BACnet-Simple ACK-PDU
8. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
   WAIT RCT

9. VERIFY System_Status != DOWNLOAD_IN_PROGRESS

BTL-15.2p-3: Unicast I-HAVE Test Changes [BTLWG-490]

Overview:
The BACnet standard (as per addendum 135-2012ar-5) now allows the IUT to send and receive a unicast responses.

Changes:

[In BTL Specified Tests, Change tests 9.32.1.1 through 9.32.1.12, as shown.]

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.*

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. 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.*

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. 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.*

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. 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)~~

9.32.1.5 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.*

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. 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.*

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. 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.*

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. 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.*

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. 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.*

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. 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)~~

9.32.1.11 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.*

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. 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 ~~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. RECEIVE
   ~~DESTINATION~~*DA* = LOCAL BROADCAST | GLOBAL BROADCAST | *TD*,
   I-Have-Request,
   'Device Identifier' = (the IUT's Device object),
   'Object Identifier' = O1,
   'Object Name' = V2

[In BTL Specified Tests, change these three tests as shown]

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. RECEIVE
   ~~DESTINATION~~*DA* = LOCAL BROADCAST | GLOBAL BROADCAST | *TD*,
   I-Have-Request,
   'Device Identifier' = (the IUT's Device object),
   'Object Identifier' = O2,
   'Object Name' = V1

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.*

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. RECEIVE
   DESTINATION = GLOBAL BROADCAST | REMOTE BROADCAST (to the network *X*~~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.*

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. RECEIVE
DA = GLOBAL BROADCAST | REMOTE BROADCAST (to the network *X*~~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)~~

[Change tests 8.32.1 through 8.32.4 already in BTL Specified Tests, by removing Protocol Revision check]

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. *TRANSMIT*
DESTINATION = IUT | 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. *TRANSMIT*
   DESTINATION = IUT | 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.

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 allowance for Unicast I-Have is added.*

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: 10 <= 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. *TRANSMIT*
   DESTINATION = IUT | 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 allowance for Unicast I-Have is added.*

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: 10 <= 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. *TRANSMIT*
   DESTINATION = IUT | 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.

BTL-15.2p-4: Tests for GW-EO-B [BTLWG-617]

Overview:

Adds testing for the new Gateway - Embedded Objects - B BIBB.

Changes:

[In BTL Checklist, modify the existing Gateway - Embedded Objects - B section]

Gateway-Embedded Objects - B		
	R[1]	Base Requirements
	O	*Supports command prioritization*
[1]~~Contact BTL for interim tests for this BIBB.~~		

[In BTL Test Plan, modify existing Gateway - Embedded Objects - B tests in section 11.2]

11 Gateway

11.2 Gateway-Embedded Objects - B

11.2.1 Base Requirements

~~Contact BTL for Interim tests for this BIBB.~~

Base requirements must be met by any IUT that claims GW-EO-B.

BTL - 9.18.1.X8 - ReadProperty Service when Non-BACnet Device Offline		
	Test Conditionality	*Must be executed.*
	Test Directives	*The test shall be conducted upon an object which is representing information arriving through a Gateway.*
	Testing Hints	
BTL - 9.20.1.X9 - ReadPropertyMultiple Service when Non-BACnet Device Offline		
	Test Conditionality	*If IUT does not support ReadPropertyMultiple service then this test shall be skipped.*
	Test Directives	*The test shall be conducted upon an object which is representing information arriving through a Gateway.*
	Testing Hints	
BTL - 9.21.1.X10 - ReadRange Service when Non-BACnet Device Offline		
	Test Conditionality	*If IUT does not support ReadRange service then this test shall be skipped. If IUT supports the ReadRange service but does not support a list property that maps onto data from a non-BACnet device, this test shall be skipped.*
	Test Directives	*The test shall be conducted upon an object which is representing information arriving through a Gateway.*
	Testing Hints	

11.2.2 Supports Command Prioritization

Gateways are required to implement Priority_Array properties correctly with all 16 entries

135.1-2013 - 7.3.1.2 - Relinquish Default Test		
	Test Conditionality	*Must be executed.*
	Test Directives	*The test shall be conducted upon an object which is representing information arriving through a Gateway. If no object can be made to meet the configuration requirements, this test shall be skipped.*
	Testing Hints	
135.1-2013 - 7.3.1.3 - Command Prioritization Test		
	Test Conditionality	*Must be executed.*
	Test Directives	*The test shall be conducted upon an object which is representing information arriving through a Gateway.*

	Testing Hints	

[In BTL Specified Test add three new tests as shown, each appended to the section for tests of the service.]

9.18.1.X8 ReadProperty Service when Non-BACnet Device Offline

Purpose: To verify that the ReadProperty service executes successfully when a non-BACnet device is offline.

Test Concept: Object1 is an object which contains information from a non-BACnet device. The non-BACnet device is verified to be online and recognized by the IUT. It is then made to go offline, and the IUT is made to recognize that the device is offline. A property, P1, from Object1 which contains a dynamic value derived from the data in the non-BACnet device is read from the IUT.

Test Steps:

1. CHECK (any vendor-specified indication, that the non-BACnet device is online)
2. MAKE (the non-BACnet device go offline)
3. MAKE (the IUT notice that the non-BACnet device is offline)
4. TRANSMIT ReadProperty Request,
   'Object Identifier' = Object1,
   'Property Identifier' = P1
5. RECEIVE ReadProperty-ACK,
   'Object Identifier' = Object1,
   'Property Identifier' = P1,
   'Property Value' = (V, any valid value)

9.20.1.X9 ReadPropertyMultiple Service when Non-BACnet Device Offline

Purpose: To verify that the ReadPropertyMultiple service executes successfully when a non-BACnet device is offline.

Test Concept: Object1 is an object which contains information from a non-BACnet device. The non-BACnet device is verified to be online and recognized by the IUT. It is then made to go offline, and the IUT is made to recognize that the device is offline. A property, P1, from Object1 which contains a dynamic value derived from the data in the non-BACnet device is read from the IUT.

Test Steps:

1. CHECK (any vendor-specified indication, that the non-BACnet device is online)
2. MAKE (the non-BACnet device go offline)
3. MAKE (the IUT notice that the non-BACnet device is offline)
4. TRANSMIT ReadPropertyMultiple-Request,
   'Object Identifier' = Object1,
   'Property Identifier' = P1
5. RECEIVE ReadPropertyMultiple-ACK,
   'Object Identifier' = Object1,
   'Property Identifier' = P1,
   'Property Value' = (any valid value)

9.21.1.X10 ReadRange Service when Non-BACnet Device Offline

Purpose: To verify that the ReadRange Service executes successfully when a non-BACnet device is offline.

Test Concept: Object1 is an object which contains information from a non-BACnet device. The non-BACnet device is verified to be online and recognized by the IUT. It is then made to go offline, and the IUT is made to recognize that the device is offline. A property, P1, from Object1 which contains a dynamic value derived from the data in the non-BACnet device is read from the IUT.

Test Steps:

1. CHECK (any vendor-specified indication, that the non-BACnet device is online)
2. MAKE (the non-BACnet device go offline)
3. MAKE (the IUT notice that the non-BACnet device is offline)
4. TRANSMIT ReadRange-Request,
   'Object Identifier' = Object1,
   'Property Identifier' = P1
5. RECEIVE ReadRange-ACK,
   'Object Identifier' = Object1,
   'Property Identifier' = P1,
   'Property Value' = (any valid value)

BTL-15.2p-5: FAULT_OUT_OF_RANGE Testing [BTLWG-297]

Overview:

Add new tests for event fault algorithm FAULT_OUT_OF_RANGE in AE-N-I-B and AE-N-E-B with a conditionality which states that, 'Must be executed if the device claims conformance to protocol revision 16 or higher'.

Changes:

[In BTL Checklist, removed the conformance codes 5, for Fault Out of Range algorithm]

Alarm and Event Management - Notification - Internal - B		
…		
	C[3,~~5~~,6]	Implements the CHANGE_OF_RELIABILITY - FAULT_OUT_OF_RANGE algorithm

[1] Required if EventNotifications with service parameter AckRequired = True can be issued.
[2] At least one of these options must be supported to claim support for this BIBB.
[3] At least one of these options must be supported to claim support for this BIBB. It is recommended that a standard BACnet algorithm be used instead of a proprietary algorithm whenever possible.
[4] At least one of these options must be supported to claim support for this BIBB. The BACnetDateTime form of the timestamp is the recommended option.
[5] Contact BTL for interim tests for this algorithm.
[6] Protocol_Revision 16 or higher must be claimed.
[7] Protocol_Revision 17 or higher must be claimed.
[8] Protocol_Revision 18 or higher must be claimed.

[In BTL Checklist, removed the conformance codes 2 and 3, for Fault Out of Range algorithm]

Alarm and Event Management - Notification - External - B		
…		
	C[1,~~2,~~3]	Implements the CHANGE_OF_RELIABILITY - FAULT_OUT_OF_RANGE algorithm
…		

[1] One of these options must be supported to claim support for this BIBB. It is recommended that a standard BACnet algorithm be used instead of a proprietary algorithm whenever possible.
[2] Contact BTL for interim tests for this algorithm.
[3] Protocol_Revision 16 or higher must be claimed.
[4] Protocol_Revision 17 or higher must be claimed.
[5] Protocol_Revision 18 or higher must be claimed.

[In BTL Test Plan add contents in section 5.2.42]

5.2 Alarm and Event Management - Notification - Internal - B

5.2.42 Implements the CHANGE_OF_RELIABILITY - FAULT_OUT_OF_RANGE algorithm
~~Contact the BTL for interim tests for this algorithm.~~

The IUT contains, or can be made to contain, an object that can generate EventNotifications with an Event_Type of CHANGE_OF_RELIABILITY and supports the specified algorithm.

BTL - 8.4.X9.15 - CHANGE_OF_RELIABILITY with the FAULT_OUT_OF_RANGE Algorithm (ConfirmedEventNotification)		
	Test Conditionality	*Must be executed*
	Test Directives	*This test shall be executed for different Reliability transitions which are supported in the IUT such as No_Fault_Detected -Under_Range, Under_Range - No_Fault_Detected, No_Fault_Detected - OverRange, Over_Range - No_Fault_Detected, No_Fault_Detected - Under_Range, Under_Range - Over_Range, Over_Range - Under_Range and Under_Range - No_Fault_Detected.*

		This test must be repeated once for each object type that is capable of generating event notifications with an Event_Type of CHANGE_OF_RELIABILITY with the FAULT_OUT_OF_RANGE
	Testing Hints	
BTL - 8.5.X9.15 - CHANGE_OF_RELIABILITY with the FAULT_OUT_OF_RANGE Algorithm (UnconfirmedEventNotification)		
	Test Conditionality	*Must be executed*
	Test Directives	*This test shall be executed for different Reliability transitions which are supported in the IUT such as No_Fault_Detected -Under_Range, Under_Range - No_Fault_Detected, No_Fault_Detected - OverRange, Over_Range - No_Fault_Detected, No_Fault_Detected - Under_Range, Under_Range - Over_Range, Over_Range - Under_Range and Under_Range - No_Fault_Detected.* *This test must be repeated once for each object type that is capable of generating event notifications with an Event_Type of CHANGE_OF_RELIABILITY with the FAULT_OUT_OF_RANGE*
	Testing Hints	

[In BTL Test Plan add contents in section 5.3.30]

5.3 Alarm and Event Management - Notification - External - B
5.3.30 Implements the CHANGE_OF_RELIABILITY - FAULT_OUT_OF_RANGE algorithm
~~Contact the BTL for interim tests for this algorithm.~~

The IUT contains, or can be made to contain, an Event Enrollment object that can generate EventNotifications with an Event_Type of CHANGE_OF_RELIABILITY and supports the specified algorithm.

BTL - 8.4.X9.15 - CHANGE_OF_RELIABILITY with the FAULT_OUT_OF_RANGE Algorithm (ConfirmedEventNotification)		
	Test Conditionality	*Must be executed*
	Test Directives	*This test shall be executed with an Event Enrollment object that is configured to monitor a property in a device other than the IUT.* *This test shall be executed for different Reliability transitions which are supported in the IUT such as No_Fault_Detected -Under_Range, Under_Range - No_Fault_Detected, No_Fault_Detected - OverRange, Over_Range - No_Fault_Detected, No_Fault_Detected - Under_Range, Under_Range - Over_Range, Over_Range - Under_Range and Under_Range - No_Fault_Detected. .* *This test must be repeated once for each object type that is capable of generating event notifications with an Event_Type of CHANGE_OF_RELIABILITY with the FAULT_OUT_OF_RANGE*
	Testing Hints	
BTL - 8.5.X9.15 - CHANGE_OF_RELIABILITY with the FAULT_OUT_OF_RANGE Algorithm (UnconfirmedEventNotification)		
	Test Conditionality	*Must be executed*
	Test Directives	*This test shall be executed with an Event Enrollment object that is configured to monitor a property in a device other than the IUT.* *This test shall be executed for different Reliability transitions which are supported in the IUT such as No_Fault_Detected -Under_Range, Under_Range - No_Fault_Detected, No_Fault_Detected - OverRange, Over_Range - No_Fault_Detected, No_Fault_Detected - Under_Range, Under_Range - Over_Range, Over_Range - Under_Range and Under_Range - No_Fault_Detected. .*

		This test must be repeated once for each object type that is capable of generating event notifications with an Event_Type of CHANGE_OF_RELIABILITY with the FAULT_OUT_OF_RANGE .
	Testing Hints	

[In BTL Specified Tests add 2 new test cases]

8.4.X9.15 CHANGE_OF_RELIABILITY with the FAULT_OUT_OF_RANGE Algorithm

Purpose: To verify the correct operation of the FAULT_OUT_OF_RANGE event algorithm.

Test Concept: Select a fault detecting object O1 which is configured to use the FAULT_OUT_OF_RANGE algorithm. Ensure that no other fault conditions exist in the object. Set pMonitoredValue to outside the range of values considered to be normal for the object. Verify the correct transition is generated. The fault condition is then removed. It is verified that O1 generates the correct notifications.

Test Configuration: O1 is configured to detect and report faults, to have no fault conditions present. The Issue_Confirmed_Notifications property shall have a value of TRUE. The event-generating objects shall be in a NORMAL state at the start of the test.

Test Steps:

1. VERIFY pCurrentReliability = NO_FAULT_DETECTED
2. VERIFY pCurrentState = NORMAL
3. IF (pMonitoredValue is writable) THEN
   WRITE pMonitoredValue = (a value less than pMinimumNormalValue | a value greater than pMaximumNormalValue)
   ELSE
   MAKE (pMonitoredValue = a value less than pMinimumNormalValue | a value greater than pMaximumNormalValue)
4. BEFORE **Notification Fail Time**,
   RECEIVE ConfirmedEventNotification-Request,
   'Process Identifier' = (any valid process identifier),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = O1,
   'Time Stamp' = (any valid time stamp),
   'Notification Class' = (the class corresponding to the object O1 being tested),
   'Priority' = (the value configured to correspond to a TO_FAULT transition),
   'Event Type' = CHANGE_OF_RELIABILITY,
   'Message Text' = (optional, any valid message text),
   'Notify Type' = EVENT | ALARM,
   'AckRequired' = TRUE | FALSE,
   'From State' = NORMAL,
   'To State' = FAULT,
   'Event Values' = (pCurrentReliability, pStatusFlags, pMonitoredValue, pMinimumNormalValue | pMaximumNormalValue)
5. TRANSMIT BACnet-SimpleACK-PDU
6. VERIFY pCurrentReliability = UNDER_RANGE | OVER_RANGE
7. VERIFY pCurrentState = FAULT
8. VERIFY pStatusFlags = (TRUE, TRUE,?,?)
9. IF (pMonitoredValue is writable) THEN
   WRITE pMonitoredValue = (a value greater than or equal to pMinimumNormalValue, and pMonitoredValue is less than or equal to pMaximumNormalValue)
   ELSE
   MAKE (pMonitoredValue = a value, greater than or equal to pMinimumNormalValue, and pMonitoredValue is less than or equal to pMaximumNormalValue)
10. BEFORE **Notification Fail Time**,
    RECEIVE ConfirmedEventNotification-Request,
    'Process Identifier' = (any valid process identifier),

'Initiating Device Identifier' =	IUT,
'Event Object Identifier' =	O1,
'Time Stamp' =	(any valid time stamp),
'Notification Class' =	(the class corresponding to the object O1 being tested),
'Priority' =	(the value configured to correspond to a TO_FAULT transition),
'Event Type' =	CHANGE_OF_RELIABILITY,
'Message Text' =	(optional, any valid message text),
'Notify Type' =	EVENT | ALARM
'AckRequired' =	TRUE | FALSE,
'From State' =	FAULT,
'To State' =	NORMAL,
'Event Values' =	(pCurrentReliability, pStatusFlags, pMonitoredValue, pMinimumNormalValue | pMaximumNormalValue)

11. TRANSMIT BACnet-SimpleACK-PDU
12. VERIFY pCurrentReliability = NO_FAULT_DETECTED
13. VERIFY pCurrentState = NORMAL
14. VERIFY pStatusFlags = (FALSE, FALSE,?, ?)

8.5.X9.15 CHANGE_OF_RELIABILITY with the FAULT_OUT_OF_RANGE Algorithm

Purpose: To verify the correct operation of the FAULT_OUT_OF_RANGE event algorithm.

Test Concept: Select a fault detecting object O1 which is configured to use the FAULT_OUT_OF_RANGE algorithm. Ensure that no other fault conditions exist in the object. Set pMonitoredValue to outside the range of values considered to be normal for the object. Verify the correct transition is generated. The fault condition is then removed. It is verified that O1 generates the correct notifications.

Test Configuration: O1 is configured to detect and report faults, to have no fault conditions present. The Issue_Confirmed_Notifications property shall have a value of TRUE. The event-generating objects shall be in a NORMAL state at the start of the test.

Test Steps: The test steps for this test case are identical to the test steps in 8.4.X9.15 except that the ConfirmedEventNotification requests are UnconfirmedEventNotification requests and the TD does not acknowledge receiving the notifications.

Notes to Tester: The passing results for this test case are identical to the ones in 8.4.X9.15 except that the event notifications shall be conveyed using an UnconfirmedEventNotification service request. The MAC address used for these messages shall be either a broadcast that reaches the local network of the TD or the MAC address of the TD.

BTL-15.2p-6: Offline Devices on a Virtual Network [BTLWG-176]

Overview:
A router that routes to virtual networks only, can be active when one or more virtual devices to which it routes goes offline. GW-VN-B specifies that a router in this situation must not respond to incoming requests concerning the offline virtual device.

A new test is added to verify this behavior.

Changes:
[In the BTL Test Plan, add a new test, to be executed in the Base Requirements of Virtual Router. Ensure that the new test is inserted in the middle of section amongst existing tests, to preserve that all tests in section appear in increasing numeric order.]

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

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

...		
BTL - 10.8.3.6.X1 - Silently Drop Messages to a Virtual Device that is Offline		
	Test Conditionality	*Must be executed if any virtual device can become offline, for a time.*
	Test Directives	
	Testing Hints	

...

[In BTL Specified Tests, add new test in new sub-section 10.8.3.6, as shown.]

10.8.3.6.X1 Silently Drop Messages to a Virtual Device that is Offline
Purpose: To verify that the IUT does not return any message in response to an NPDU with a destination that is offline.

Test Concept: The non-BACnet device is verified to be online and recognized by the IUT. It is then made to go offline, and the IUT is made to recognize that the device is offline. A property, P1, from Object1 which is derived from the data in a virtual device is read from the IUT. Verify that when a virtual device is off-line, that the IUT sends no response to messages that are directed to that off-line device.

Configuration Requirements: The IUT acting as a virtual router, shall be configured so that a virtual device VD1A which can sometimes be online, is initially online for this test. If no virtual device can become off-line, then this test shall be skipped.

Test Steps:

1. CHECK (any vendor-specified indication, that the virtual device is online)
2. MAKE (the virtual device containing Object1 go offline)
3. MAKE (the IUT notice that the virtual device is offline)
4. TRANSMIT ReadProperty-Request,
   'Object Identifier' = Object1,
   'Property Identifier' = P1
5. CHECK (that no responsive message is returned from IUT)
6. TRANSMIT PORT A, // in defined in Figure 10-4 attached at Network 2
   DESTINATION = VD1A,
   Message Type = (any valid value)
7. CHECK (that no responsive message is returned from IUT)

BTL-15.2p-7: Add Testing to Enforce J.2.1.2 [BTLWG-488]

Overview:

The current test package does not currently enforce the behavior specified in clause J.2.12 which states that an IUT shall not use original-broadcast NPDU's when registered as a foreign device.

Changes:

[In BTL Test Plan, sections 9.3.3 and 9.4.7, change test reference for 14.9.1 to BTL test 14.9.1]

~~135.1-2013 - 14.9.1~~ *BTL - 14.9.1* - **Distribute-Broadcast-To-Network**		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	

[In BTL Specified Tests, add the following test (derived from 135.1-2013 - 14.9.1)]

14.9.1 Distribute-Broadcast-To-Network

Reason for Change: Modified test to enforce the behavior specified in clause J.2.12.

Dependencies: 14.8, “Registering as a Foreign Device”

BACnet Reference Clause: J.2.10

Purpose: This test case verifies that the IUT, registered as a foreign device, can issue a request to a BBMD to broadcast the message on all subnets in the BBMD’s BDT.

Test Concept: The IUT is configured to register itself as a foreign device with the TD, then after registration is achieved it is caused to initiate a broadcast message to be conveyed to the BBMD for distribution. If the IUT does not support foreign device registration, or cannot initiate broadcast messages conveying a BACnet NPDU, then this test shall be omitted.

Test Steps:

1. RECEIVE DESTINATION = TD, SOURCE = IUT,
   Register-Foreign-Device
2. TRANSMIT DESTINATION = IUT, SOURCE = TD,
   BVLC-Result,
   ‘Result Code’ = Successful completion
3. MAKE (*a condition that would make* the IUT *normally* initiate a broadcast)
4. RECEIVE DESTINATION = TD, SOURCE = IUT,
   Distribute-Broadcast-To-Network
5. *CHECK (that the IUT does not transmit an Original-Broadcast-NPDU on this port)*

BTL-15.2p-8: Updated Testing Hints for Reading of NULL [BTLWG-114]

Overview:

The BACnet standard now allows in addition to Schedule_Default, the Alarm_Values and the Fault_Values of the CharacterString Value Object as well as property Low_Diff_Limit in the Loop Objects are standard properties that should accept a written NULL. Added the same in Testing Hints.

Changes:

[In Test Plan modify the sections as shown]

4.1 Data Sharing - ReadProperty - A

4.1.7 Can Read NULL Property Values

The IUT is able to read NULL values.

135.1-2013 - 8.18.1 - Reading Non-Array Properties		
	Test Conditionality	This test can be skipped if 8.18.2 is executed against a property that contains a NULL value.
	Test Directives	~~The property that the IUT reads shall contain a NULL value.~~
	Testing Hints	Non-array propert~~y~~*ies* that might contain a NULL value ~~is~~ *are* the Present_Value, *Schedule_Default* propert~~y~~*ies* of a Schedule object, *and Low_Diff_Limit in the Loop Objects*
135.1-2013 - 8.18.2 - Reading an Array Element		
	Test Conditionality	This test can be skipped if 8.18.1 is executed against a property that contains a NULL value.
	Test Directives	~~The property that the IUT reads shall contain a NULL value.~~
	Testing Hints	*Array properties that might contain a NULL value are the Alarm_Values and Fault_Values of the CharacterString Value Object*

4.2 Data Sharing - ReadProperty - B

4.2.7 Contains NULL Property Values

The IUT contains, or can be made to contain, a property with the value of NULL.

BTL - 9.18.1.X1 - Reading Properties Based on Data Type		
	Test Conditionality	If the IUT contains a property with the value NULL when test BTL - 7.1 is executed and the EPICS does not contain a ? for this property, this test can be skipped.
	Test Directives	
	Testing Hints	*Schedule_Default, Present_Value of Schedule Object, Alarm_Values and Fault_Values of the CharacterString Value Object and Low_Diff_Limit in the Loop Object are standard properties that contain NULL values.*

4.3 Data Sharing - ReadPropertyMultiple - A

4.3.17 Can Read NULL Property Values

The IUT is able to read NULL values.

135.1-2013 - 8.20.1 - Reading a Single Property of a Single Object, 135.1-2013 - 8.20.2 - Reading Multiple Properties of a Single Object, 135.1-2013 - 8.20.3 - Reading Multiple Objects, One Property Each, or 135.1-2013 - 8.20.4 - Reading Multiple Objects, Multiple Properties for Each		
	Test Conditionality	At least one of the tests (8.20.1..8.20.4) shall be executed against a property with a NULL value.
	Test Directives	~~At least one of the properties read by the selected test shall contain a NULL value.~~
	Testing Hints	*Schedule_Default, Present_Value of Schedule Object, Alarm_Values and Fault_Values of the CharacterString Value Object and Low_Diff_Limit in the Loop Object are standard properties that contain NULL values.*

4.4 Data Sharing - ReadPropertyMultiple - B

4.4.7 Contains NULL Property Values

The IUT contains, or can be made to contain, a property with the value of NULL.

BTL - 9.20.1.X1 - Reading Properties Based on Data Type		
	Test Conditionality	If the IUT contains a property with a NULL data type when test BTL - 7.1 is executed this test can be skipped.
	Test Directives	
	Testing Hints	*Schedule_Default, Present_Value of Schedule Object, Alarm_Values and Fault_Values of the CharacterString Value Object and Low_Diff_Limit in the Loop Object are standard properties that contain NULL values.*

4.5 Data Sharing - WriteProperty - A

4.5.7 Can Write NULL Property Values to non-commandable Properties

The IUT is able to write NULL values to non-commandable properties.

135.1-2013 - 8.22.1 - Writing Non-Array Properties, or 135.1-2013 - 8.22.2 - Writing Array Properties		
	Test Conditionality	At least one of these tests must be applied with the IUT writing a NULL value.
	Test Directives	
	Testing Hints	*Schedule_Default, Present_Value of Schedule Object, Alarm_Values and Fault_Values of the CharacterString Value Object and Low_Diff_Limit in the Loop Object, are standard properties that should accept a written NULL.*

[In Test Plan sections 4.6.7 add test Conditionality for BTL - 9.22.1.X2 as shown]

4.6 Data Sharing - WriteProperty - B

4.6.7 Contains non-commandable Properties which Accept a Written NULL Value

The IUT contains, or can be made to contain, a writable property that accepts a written NULL value.

BTL - 9.22.1.X2 - Writing to Properties Based on Data Type		
	Test Conditionality	If the IUT cannot be configured to contain a property that will accept and retain a NULL value, then this test shall be omitted. Note that commandable properties do not retain the NULL value and as such devices that accept NULL values only for the relinquishing of commanded values will not be subjected to this test.

	Test Directives	
	Testing Hints	*Schedule_Default, Present_Value of Schedule Object, Alarm_Values and Fault_Values of the CharacterString Value Object and Low_Diff_Limit in the Loop Object, are standard properties that should accept a written NULL.*

[In Test Plan sections 4.7.13 add test Conditionality for 135.1-2013 - 8.23.1, 135.1-2013 - 8.23.2, 135.1-2013 - 8.23.3, 135.1-2013 - 8.23.4 as shown]

4.7 Data Sharing - WritePropertyMultiple - A

4.7.13 Can Write NULL Property Values to non-commandable Properties

The IUT is able to write NULL values to non-commandable properties.

135.1-2013 - 8.23.1 - Writing a Single Property of a Single Object, 135.1-2013 - 8.23.2 - Writing Multiple Properties of a Single Object, 135.1-2013 - 8.23.3 - Writing Multiple Objects, One Property Each, or 135.1-2013 - 8.23.4 - Writing Multiple Objects, Multiple Properties for Each		
	Test Conditionality	At least one of the tests (8.23.1...8.23.4) shall be executed against a property with a NULL value.
	Test Directives	
	Testing Hints	*Schedule_Default, Present_Value of Schedule Object, Alarm_Values and Fault_Values of the CharacterString Value Object and Low_Diff_Limit in the Loop Object, are standard properties that should accept a written NULL.*

[In Test Plan sections 4.8.10 add test Conditionality for 135.1-2013 - 9.23.1.8 as shown]

4.8 Data Sharing - WritePropertyMultiple - B

4.8.10 Contains non-commandable Properties which Accept a Written NULL Value

The IUT contains, or can be made to contain, a writable property that accepts a written NULL value.

135.1-2013 - 9.23.1.8 - Writing to Properties Based on Data Type		
	Test Conditionality	If the IUT cannot be configured to contain a property that will accept and retain a NULL value, then this test shall be omitted. Note that commandable properties do not retain the NULL value and as such devices that accept NULL values only for the relinquishing of commanded values will not be subjected to this test.
	Test Directives	
	Testing Hints	*Schedule_Defaul, Present_Value of Schedule Object, Alarm_Values and Fault_Values of the CharacterString Value Object and Low_Diff_Limit in the Loop Object, are standard properties that should accept a written NULL.*

4.19 Data Sharing - Change of Value Property - A

4.19.11 Can Subscribe to NULL Property Values

The IUT can subscribe for, receive, and process Change of Value notifications from property that contains a NULL value

BTL- 8.11.X1.1 - Change of Value Notifications		
	Test Conditionality	Either a confirmed or an unconfirmed COV notification may be observed.

	Test Directives	Execute test using 'Monitored Property Identifier' = (any property which the vendor supports in a SubscribeCOVProperty-Request that can contain a NULL value)
	Testing Hints	*Schedule_Default, Present_Value of Schedule Object, Alarm_Values and Fault_Values of the CharacterString Value Object and Low_Diff_Limit in the Loop Object, are standard properties that should accept a written NULL.*

7.1 Trending - Viewing - A

7.1.3 Interoperates with Trend Logs

The IUT can interoperate with Trend Log objects.

BTL - 8.21.9 - Presents Log Records		
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test for Real, INTEGER, BOOLEAN, Bit-String, Enumerated, and NULL datatypes.
	Testing Hints	*Schedule_Default, Present_Value of Schedule Object, Alarm_Values and Fault_Values of the CharacterString Value Object and Low_Diff_Limit in the Loop Object, are standard properties that should accept a written NULL.*

7.1.4 Interoperates with Trend Log Multiple Objects

The IUT can interoperate with Trend Log Multiple objects in devices claiming Protocol_Revision 7 or higher.

BTL - 8.21.9 - Presents Log Records		
	Test Conditionality	Must be executed.
	Test Directives	Execute test against Trend Log Multiple that contains all required datatypes (Boolean, Real, Enumerated, Unsigned32, Integer32, Bit String, and NULL).
	Testing Hints	*Schedule_Default, Present_Value of Schedule Object, Alarm_Values and Fault_Values of the CharacterString Value Object and Low_Diff_Limit in the Loop Object, are standard properties that should accept a written NULL.*

7.2 Trending - Advanced View and Modify – A

7.2.1 Base Requirements

BTL - 8.21.9 - Presents Log Records		
	Test Conditionality	Must be executed.
	Test Directives	Repeat the test for Real, INTEGER, BOOLEAN, Bit-String, Enumerated, and NULL datatypes.
	Testing Hints	*Schedule_Default, Present_Value of Schedule Object, Alarm_Values and Fault_Values of the CharacterString Value Object and Low_Diff_Limit in the Loop Object, are standard properties that should accept a written NULL.*

7.4 Trending - View and Modify Trends - E - B

7.4.8 Is Able to Trend NULL Values

The IUT can be made to trend any type property that may change to a value of NULL.

135.1-2013 - 9.21.1.11 - Data Type Verification Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	*The best way to do this is to trend an entry in a priority array. Schedule_Default, Present_Value of Schedule Object, Alarm_Values and Fault_Values of the CharacterString Value Object and Low_Diff_Limit in the Loop Object, are standard properties that should accept a written NULL.*

7.7 Trending - View and Modify Multiple Values - I - B

7.7.14 Is Able to Trend NULL Datatypes

The IUT is able to trend NULL data type in a Trend Log Multiple Object.

135.1-2013 - 9.21.1.1 - Reading All Items in the List**, or** **135.1-2013 - 9.21.1.2 -** Reading Items by Position with Positive Count**, or** **135.1-2013 - 9.21.1.3 -** Reading Items by Position with Negative Count**, or** **135.1-2013 - 9.21.1.4 -** Reading Items by Time**, or** **135.1-2013 - 9.21.1.4.1 -** Reading Items by Time with Negative Count**, or** **135.1-2013 - 9.21.1.9 -** Reading Items by Sequence with Positive Count**, or** **135.1-2013 - 9.21.1.10 -** Reading Items by Sequence with Negative Count**.**		
	Test Conditionality	At least one of the tests listed above shall be run.
	Test Directives	
	Testing Hints	*Schedule_Default, Present_Value of Schedule Object, Alarm_Values and Fault_Values of the CharacterString Value Object and Low_Diff_Limit in the Loop Object, are standard properties that should accept a written NULL.*

7.8 Trending - View and Modify Multiple Values - E - B

7.8.9 Is Able to Trend NULL Datatypes

The IUT is able to trend NULL data type in a Trend Log Multiple Object.

135.1-2013 - 9.21.1.1 - Reading All Items in the List**, or** **135.1-2013 - 9.21.1.2 -** Reading Items by Position with Positive Count**, or** **135.1-2013 - 9.21.1.3 -** Reading Items by Position with Negative Count**, or** **135.1-2013 - 9.21.1.4 -** Reading Items by Time**, or** **135.1-2013 - 9.21.1.4.1 -** Reading Items by Time with Negative Count**, or** **135.1-2013 - 9.21.1.9 -** Reading Items by Sequence with Positive Count**, or** **135.1-2013 - 9.21.1.10 -** Reading Items by Sequence with Negative Count**.**		
	Test Conditionality	At least one of the tests listed above shall be run.
	Test Directives	
	Testing Hints	*Schedule_Default, Present_Value of Schedule Object, Alarm_Values and Fault_Values of the CharacterString Value Object and Low_Diff_Limit in the Loop Object, are standard properties that should accept a written NULL.*

BTL-15.2p-9: Disallow Duplicate Time Entries in Schedules [BTLWG-564]

Overview:
Add the tests for verifying that duplicate entries for TimeValue pair are not accepted for schedule object.

Changes:

[In BTL Test Plan, add reference to list addendum r in Section 1.1- External Document References]

[In BTL Test Plan, add reference to new test to the Base Requirements of Scheduling-Internal-B]

6.4 Scheduling - Internal - B

6.4.1 Base Requirements

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

…		
135.1-2013r - 7.3.2.23.X Forbid Duplicate Time Values		
	Test Conditionality	*If Protocol_Revision < 16, then this test shall be skipped.*
	Test Directives	*Apply to a writable Weekly_Schedule and then to an Exception_Schedule property of a schedule object*
	Testing Hints	

[In BTL Test Plan, add reference to new test to the Base Requirements of Scheduling-Weekly Schedule-Internal-B]

6.6 Scheduling - Weekly Schedule - Internal - B

6.6.1 Base Requirements

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

…		
135.1-2013r - 7.3.2.23.X Forbid Duplicate Time Values		
	Test Conditionality	*If Protocol_Revision < 16, then this test shall be skipped.*
	Test Directives	*Apply to a writable Weekly_Schedule property of a schedule object*
	Testing Hints	

BTL-15.2p-10: Remove COVP for Basic Proprietary Properties [BTLWG-657]

Overview:

The COVP changes in 15.2j have an entry for proprietary properties of basic datatypes but the testing is identical to that for standard properties so there is no reason to separate the handling of them. Plus it adds confusion for test labs and applicants.

Changes:

[In Addenda 15.2j-1 modify Checklist changes for COVP]

Data Sharing - Change Of Value Property - B		
	R	Base Requirements
	R	Supports COVP Lifetimes up to 8 hours in duration
	R	Supports COVP for Status_Flags changes
	C[1]	Supports COVP for non-array property
	C[1]	Supports COVP for array element
	C[1]	Supports COVP for the size of an array
	C[1]	Supports COVP for whole array
	O	Supports COVP for list property
	C[2]	Supports COVP for NULL property values
	C[2]	Supports COVP for BOOLEAN property values
	C[2]	Supports COVP for Enumerated property values
	C[2]	Supports COVP for INTEGER property values
	C[2]	Supports COVP for Unsigned property values
	C[2]	Supports COVP for REAL property values
	C[2]	Supports COVP for Double property values
	C[2]	Supports COVP for Time property values
	C[2]	Supports COVP for Date property values
	C[2]	Supports COVP for CharacterString property values
	C[2]	Supports COVP for OctetString property values
	C[2]	Supports COVP for BitString property values
	C[2]	Supports COVP for BACnetObjectIdentifier property values
	C[2]	Supports COVP for constructed property values
	~~C[2]~~	~~Supports COVP for proprietary property values of basic data types~~
[1] At least one of these options is required in order to claim conformance to this BIBB. [2] At least one of these options is required in order to claim conformance to this BIBB.		

[In Addenda 15.2j-1 remove the change to the BTL Test Plan adding in 4.20.23]

~~4.20.23 Supports COVP to Proprietary Property Values of Basic Data Types~~

~~The IUT supports change of value notifications for at least one proprietary property values of basic data types.~~

~~BTL - 9.11.1.X11 - Confirmed Change of Value Notification from Property Value~~		
	~~Test Conditionality~~	~~Must be executed.~~
	~~Test Directives~~	~~Select parameters for an object and property which supports SubscribeCOVProperty.~~ ~~*Repeat test for at least one object of each type that has at least one property which supports SubscribeCOVProperty, repeat test for all Properties which support SubscribeCOVProperty.*~~
	~~Testing Hints~~	
~~BTL - 9.11.1.X12 - Unconfirmed Change of Value Notification from Property Value~~		
	~~Test Conditionality~~	~~Must be executed.~~
	~~Test Directives~~	~~Apply the test to an object and property which supports SubscribeCOVProperty.~~ ~~*Repeat test for at least one object of each type that has at least one property which supports SubscribeCOVProperty, repeat test for all Properties which support SubscribeCOVProperty*~~
	~~Testing Hints~~	