BACnet® TESTING LABORATORIES
ADDENDA

Addendum imp1 to
BTL Test Package 20.0.1

Revision v6
Revised 11/1/2022

Approved by the BTL Working Group on 2022-10-03.
Approved by the BTL Working Group Voting Members on 2022-10-31.
Published on 2022-11-02.

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

FOREWORD

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

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

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

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

BTL-20.0.1 imp1-1: Add MSTP Router Reply Postponed Test [BTLWG-581]

Overview:

Add MSTP Router Reply Postponed Test

Changes:

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, update section 9.1 Data Link Layer - MS/TP - Master Node]

9.1.6 Is a BACnet Router

The IUT is or can be configured to act as a BACnet router.

135.1-2019 - 12.1.3.15 - Sole Master Test		
	Test Conditionality	~~Only one device on the MS/TP network~~*Must be executed.*
	Test Directives	*Only one master on the MS/TP network.*
	Testing Hints	
BTL - 12.1.X - MS/TP Router Reply Postponed Test		
	Test Conditionality	*Must be executed.*
	Test Directives	*Apply for BACnet Data Expecting Reply Frame Type.* *For devices at Protocol_Revision 16 or greater, repeat for BACnet Extended Data Expecting Reply Frame Type*
	Testing Hints	

Test Changes

[Insert new Test]

12.1.X MS/TP Router Reply Postponed Test

Reference: 9.7.1

Purpose: To verify that the IUT sends 'reply postponed' when it receives a MS/TP packet with a data_expecting_reply parameter set to TRUE destined for another network.

Test Concept: The IUT is configured to route between Network 1 on port A (MS/TP network) and Network 2 on port B (any BACnet data link). D1A resides on Network 1 and D2C resides on Network 2. The IUT receives a message from D1A destined for D2C with the data expecting reply parameter set to TRUE. The IUT transmits ‘Reply Postponed’ to D1A before attempting to route the message.

Configuration Requirements: The IUT is actively routing between Network 1 and Network 2 and D1A has discovered D2C.

Test Steps:

1. TRANSMIT PORT A
    SA = D1A
    DA = IUT
    DNET = Network 2
    DADR = D2C
    Hop Count = 255
    BACnet-Confirmed-Request-PDU
2. RECEIVE PORT A

Reply Postponed

3. RECEIVE PORT B
    SA = IUT
    DA = D2C
    SNET = Network 1
    SADR = D1A
    Hop Count = (any integer x: 0 < x < 255)
    BACnet-Confirmed-Request-PDU

BTL-20.0.1 imp1-2: Clarify the Testing for the Event_Algorithm_Inhibit Test [BTLWG-745]

Overview:

Jira item BTLWG-745. The test conditionality and test concept for this test currently does not covers the case where it needs to be executed against an object which contains an Event_Algorithm_Ref property, but that property is not initialized.

Changes:

Checklist Changes

None

Test Plan Changes

[Change 5.2.1, entry for 7.3.1.19.1 - Event_Algorithm_Inhibit_Test]

~~135.1-2019~~ *BTL* - 7.3.1.19.1 - Event_Algorithm_Inhibit Test		
	Test Conditionality	Apply this test when the Event_Algorithm_Inhibit property is present in an object which does not support the Event_Algorithm_Inhibit_Ref property *or where the Event_Algorithm_Inhibit_Ref property is not initialized*. If the IUT cannot be configured to contain such an object, then this test shall be skipped.
	Test Directives	The object types selected by the tester should include all variants that differ in the set of supported alarming properties, or the writability of any of those properties. At least one instance of each variant should be selected.
	Testing Hints	

[Change 5.22.1, entry for 7.3.1.19.1 - Event_Algorithm_Inhibit_Test]

~~135.1-2019~~ *BTL* - 7.3.1.19.1 - Event_Algorithm_Inhibit Test		
	Test Conditionality	~~If the IUT has no object in which the Event_Algorithm_Inhibit property is present and does not support the Event_Algorithm_Inhibit_Ref property, or has no object in which Event_Detection_Enable can be made TRUE, this test shall be skipped. If the IUT cannot be configured to contain any object capable of an event transition, then this test shall be skipped~~. *If the IUT has no object which generates CHANGE_OF_LIFE_SAFETY notifications in which the Event_Algorithm_Inhibit property is present and does not support the Event_Algorithm_Inhibit_Ref property or where the Event_Algorithm_Inhibit_Ref property is not initialized, then this test shall be skipped.*
	Test Directives	The object types selected by the tester should include all variants that differ in the set of supported alarming properties, or the writability of any of those properties. At least one instance of each variant should be selected.
	Testing Hints	

[Change 5.32.1, entry for 7.3.1.19.1 - Event_Algorithm_Inhibit_Test]

~~135.1-2019~~ *BTL* - 7.3.1.19.1 - Event_Algorithm_Inhibit Test		
	Test Conditionality	If the IUT has no object which generates ACCESS_EVENT notifications in which the Event_Algorithm_Inhibit property is present and does not support the Event_Algorithm_Inhibit_Ref property *or*

		where the Event_Algorithm_Inhibit_Ref property is not initialized, then this test shall be skipped.
	Test Directives	Apply to an object which generates ACCESS_EVENT notifications.
	Testing Hints	

Test Changes

[Modify 7.3.1.19.1 and move into BTL Specified Tests]

7.3.1.19.1 Event_Algorithm_Inhibit Test

Reason For Change: The Test concept needs to include some language for a case where the test also needs to be executed against an event initiating object where Event_Algorithm_Inhibit_Ref property is not initialized.

Purpose: To verify that Event_Algorithm_Inhibit property in objects with intrinsic or algorithmic reporting controls whether or not the event state detection algorithm is executed.

Test Concept: Select an event generating object, O1, which supports the Event_Algorithm_Inhibit property and does not support the Event_Algorithm_Inhibit_Ref property *or where the Event_Algorithm_Inhibit_Ref property is present and not initialized*. With Event_Algorithm_Inhibit set to FALSE, make a condition exist that should result in an event transition to a normal or offnormal state. Verify that a transition occurs and that a notification is generated. Set Event_Algorithm_Inhibit to TRUE. If not already in a NORMAL state, verify that the object transitions to NORMAL. Make a condition exist that should result in an event transition, if the object Event_Algorithm_Inhibit were FALSE. If O1 supports fault detection, make a fault condition exist and verify that object detects the fault and transitions to FAULT.

Configuration Requirements: O1 is configured to detect and report unconfirmed events, is in the NORMAL state, and, if supported, is configured to detect fault conditions. D1 is either the pTimeDelay, or pTimeDelayNormal parameter, or 0 (for transitions to and from FAULT state) depending on the event transition.

Test Steps:

[No Changes]

BTL-20.0.1 imp1-3: Update the EPICS Consistency Tests to Validate Property Ranges [BTLWG-849]

Overview:

Update the EPICS Consistency Tests to validate property ranges.

Changes:

Checklist Changes

None

Test Plan Changes

None

Test Changes

5. EPICS CONSISTENCY TESTS

Reason for Change: Improved the language in this set of tests to clarify the exact requirement of the test.

These tests are static tests of the EPICS and do not involve interrogating the IUT. There are no Test Configuration or Test Step sections with TCSL in these tests because the tests are static tests of the EPICS and not tests of the IUT itself.

Each implementation shall be tested to ensure consistency among interrelated data elements. These tests shall include:

(a) All object types required by the specified BIBBs shall be indicated as supported in the Standard Object Types Supported section of the EPICS.

(b) A minimum of one instance of each object type required by the specified BIBBs shall be included in the test database.

(c) The Protocol_Object_Types_Supported property of the Device object in the test database shall indicate support for each object type required by the supported BIBBs.

(d) All application services required by the supported BIBBs shall be indicated as supported in the BACnet Standard Application Services Supported section of the EPICS with Initiate and Execute indicated as required by the supported BIBBs.

(e) The Protocol_Services_Supported property of the Device object in the test database shall indicate support for each application service for which the supported BIBBs requires support for execution of the service.

(f) The object types listed in the Standard Object Types Supported section of the EPICS shall have a one-to-one correspondence with object types listed in the Protocol_Object_Types_Supported property of the Device object contained in the test database.

(g) For each object type listed in the Standard Object Types Supported* section of the EPICS there shall be at least one object of that type in the test database. It is permissible for there to be no instance of the File object type if File objects are dynamically creatable and come into existence only temporarily during Backup and restore.
**An object type is supported if it can be made to exist in the IUT's database.*

(h) There shall be a one-to-one correspondence between the objects listed in the Object_List property of the Device object and the objects included in the test database. The Object_List property and the test database shall both include all proprietary objects. Properties of proprietary objects that are not required by BACnet Clause 23.4.3 need not be included in the test database.

(i) For each object included in the test database, all required properties for that object as defined in Clause 12 of BACnet shall be present. Standard properties which are not defined for the implemented Protocol_Revision shall not be present. In addition, if any of the properties supported for an object require the conditional presence of other properties, their presence shall be verified.

(j) For each property that is required to be writable, or conditionality writable, that property shall be marked as writable, or conditionality writable, in the EPICS.

(k) The length of the Protocol_Services_Supported bitstring shall have the number of bits defined for BACnetProtocolServicesSupported for the IUT's declared protocol revision.

(l) The length of the Protocol_Object_Types_Supported bitstring shall have the number of bits defined for BACnetObjectTypesSupported for the IUT's declared protocol revision

(m) For each object included in the test database, any properties that are deprecated or removed shall not appear after the Protocol_Revision in which the property was deprecated or removed.

(n) If the Protocol_Revision property is present in the Device object and its value is greater than or equal to 14, the Property_List property of each object included in the test database shall have one entry for each property present including non-standard properties with the exception of Object_Type, Object_Identifier, Object_Name and Property_List

(o) If the Segmentation_Supported property in the Device object is SEGMENTED_BOTH or SEGMENTED_RECEIVE, then the value of the Max_Segments_Accepted property of the Device object shall be greater than 1.

(p) For each property that is required to be read-only, that property shall not be marked as writable, or conditionality writable, in the EPICS.

(q) For each property for which the standard limits the value range, the value for the property in the EPICS, if provided, shall be within the allowable range as defined in the property definition.

(r) For each property which has a restricted value range defined in the EPICS, the restricted value range shall be within the allowable value range and contain the minimal value range as defined by the standard.

BTL-20.0.1 imp1-4: Apply Mandate for NM-BBMC-B to Test Package [BTLWG-1047]

Overview:

Apply mandate for NM-BBMC-B to the test package.

Changes:

Checklist Changes

[In BTL Checklist, modify checklist Data Link Layer – IPv4 and IPv6]

Data Link Layer - IPv4		
	R	Base Requirements
	C[1]	Is able to operate in Normal mode
	C[1]	Is able to operate in Foreign mode
	C[2]	Is able to operate in BBMD mode
	O	Supports configuration through Network Port object
	O	Is able to initiate broadcast messages
	O	Supports Network Port objects and DHCP
	O	Supports Network Address Translation in BBMD mode
	BTL-C[3]	*Supports NM-BBMDC-B*
[1] Required if the device does not support BBMD mode. [2] Required if the device does not support Foreign mode. *[3] Required if the device is able to operate in BBMD mode*		

Data Link Layer - IPv6		
	R	Base Requirements
	C[1]	Is able to operate in Normal mode
	C[1]	Is able to operate in Foreign mode
	C[2]	Is able to operate in BBMD mode
	R	Supports configuration through Network Port object
	O	Supports DHCP
	BTL-C[3]	*Supports NM-BBMDC-B*
[1] Required if the device does not support BBMD mode. [2] Required if the device does not support Foreign mode. *[3] Required if the device is able to operate in BBMD mode*		

Test Plan Changes

[In BTL Test plan, add entries for the 'Supports NM-BBMDC-B and change test directives for test BTL-14.X10.2]

9.3.9 Supports NM-BBMDC-B

The IUT claims support for NM-BBMDC-B

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify the IUT claims support for NM-BBMDC-B
	Testing Hints	

9.8.7 Supports NM-BBMDC-B

The IUT claims support for NM-BBMDC-B

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify the IUT claims support for NM-BBMDC-B
	Testing Hints	

10.7.2 Supports Registration by Foreign Devices

While configured as a BBMD, the IUT supports, or can be made to support, registration by Foreign Devices and forwards as original BACnet/IP unicasts to each, any broadcasts it processes.

BTL - 14.X10.2 - Holds at least 5 Foreign Device Registrations		
	Test Conditionality	Must be executed.
	Test Directives	*Execute the test such that one Foreign Device has a lifetime of 30s and another Foreign Device has a lifetime of 32400s (9 hours).*
	Testing Hints	
BTL - 14.X10.3 - Negative Foreign Device Registration when BBMD_Accept_FD_Registrations is FALSE		
	Test Conditionality	If the device claims Protocol_Revision 16 or lower, this test shall be skipped.
	Test Directives	
	Testing Hints	

Test Changes

None

BTL-20.0.1 imp1-5: Fix Tests to Remove Assumption of Writable Status_Flags Property [BTLWG-1181]

Overview:

Given that Status_Flags is a reflection of other properties, it is not a property that should ever be writable. Any test that attempts to, or infers that writing might be allowed, should be changed to not do so.

Changes:

Checklist Changes

None

Test Plan Changes

None

Test Changes

[Modify test 8.2.2 as shown]

8.2.2 Change of Value Notification for Changes to Status_Flags Property
~~**8.2.2 Change of Value Notification from an Analog Input, Analog Output, and Analog Value Object Status_Flags Property**~~
Reason for Change: Add more primitive value objects. Updated 'Configuration Requirements'. Removed extraneous SimpleACKs after WRITE statements. Updated descriptive text for 'List of Value' property. Removed assumption that Status_Flags is writable.

Purpose: To verify that the IUT can initiate ConfirmedCOVNotification service requests conveying a change of the Status_Flags property ~~of Analog Input, Analog Output, and Analog 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 *made to* change~~d~~ 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~~
For implementations where the Status_Flags cannot be made to change, 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*~~*. If the COV_Increment is supported or which has a writable Out_Of_Service.*~~

Test Steps:

REPEAT X = (one supported object of each type ~~from the set Analog Input, Analog Outputand Analog 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")
~~6.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 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~~

[Modify test 8.2.X10, X11, X12, X13 as shown. Note: These tests do not exist in 135.1 and therefore there will be no italic or strike through in the test when merged into the Specified Tests document. They are shown here for clarity.]

8.2.X10 ConfirmedCOVNotification Pulse Converter changing Status_Flags
Reason for Change: New Test

Purpose: To verify the correct operation of COV in the Pulse Converter object. The Pulse Converter initiates periodic COV Notifications every COV_Period, even when there are no changes in the object, in addition to the COV notifications that this object type generates due to changes in the Status_Flags property.

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~~*made to change* and a notification shall be received. ~~For some implementations writing to the Out_Of_Service property will accomplish this task.~~ For implementations ~~where it is not possible to write Out_Of_Service or change the Status_Flags by any other means,~~ *in which the Status_Flags property cannot be made to change,* this test shall be skipped. The object identifier of the Pulse Converter object being tested is designated as O1 in the test steps below.

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. COV_Period is configured high enough that is does not trigger many COV notifications during the execution of the test.

Test Steps:

1. TRANSMIT SubscribeCOV-Request,
'Subscriber Process Identifier' = (any value > 0 chosen by the TD),
'Monitored Object Identifier' = O1,
'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' = O1,
'Time Remaining' = (any value appropriate for the Lifetime selected),

```
        'List of Values' =              (the initial Present_Value, initial Status_Flags, and
                                        Update_Time)
4.  TRANSMIT BACnet-SimpleACK-PDU
5.  ~~IF (Out_OF-Service is writable) THEN~~
        ~~WRITE Out_OF-Service = TRUE~~
    ~~ELSE~~
        MAKE (Status_Flags = any value that differs from initial Status_Flags)
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' =      O1,
            'Time Remaining' =                   (any value appropriate for the Lifetime selected),
            'List of Values' =                   (the current Present_Value, new Status_Flags, and Update_Time)
7.      TRANSMIT BACnet-SimpleACK-PDU
8.  TRANSMIT SubscribeCOV-Request,
        'Subscriber Process Identifier' =     (the same value used in step 1),
        'Monitored Object Identifier' =       O1
9.  RECEIVE BACnet-SimpleACK-PDU
~~10. IF (Out_Of_Service is writable) THEN~~
        ~~WRITE Out_Of_Service = FALSE~~
```

8.2.X11 Change of Value Notification from an Access Door object Present_Value, Status_Flags and Door_Alarm_State property

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

Purpose: To verify that the IUT can initiate ConfirmedCOVNotification service requests conveying a change of the Present_Value property of Access Door 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*, Door_Alarm_State, and Status_Flags* of the monitored object *are*~~is~~ changed, and a notification shall be received. The Present_Value *and Door_Alarm_State* may be changed using the WriteProperty service or by another means. 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 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~~, or which has a writable Out_Of_Service~~. If no object has a Door_Alarm_State property, then steps ~~9,10,11~~ *11, 12, 13* shall be skipped. For implementations where it is not possible ~~to write Out_Of_Service or~~ change the Status_Flags by any other means, steps 5,6,7 shall be skipped

Test Steps:

```
REPEAT X = (one supported object of type Access Door) DO {
1.  TRANSMIT SubscribeCOV-Request,
        'Subscriber Process Identifier' =    (any value > 0chosen 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, initial Status_Flags, and
                             Door_Alarm_State if X has a Door_Alarm_State property)
4.  TRANSMIT BACnet-SimpleACK-PDU
```

```
5.  ~~IF (Out_OF-Service is writable) THEN~~
        ~~WRITE Out_OF-Service = TRUE~~
    ~~ELSE~~
        MAKE (Status_Flags = any value that differs from initial Status_Flags)
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' =              (ReportedPV=current Present_Value, new Status_Flags, and
                                    Door_Alarm_State if X has a Door_Alarm_State property)
7.  TRANSMIT BACnet-SimpleACK-PDU

8.  IF (Present_Value is writable) THEN
        WRITE X,Present_Value = (any value that differs from ReportedPV)
    ELSE
        MAKE (Present_Value = any value that differs from ReportedPV)
9.  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, new Status_Flags, and Door_Alarm_State if X has a
                                    Door_Alarm_State property)
10. TRANSMIT BACnet-SimpleACK-PDU

11. IF (Door_Alarm_State is now writable) THEN
        WRITE Door_Alarm_State = (any value that differs from its initial Door_Alarm_State)
    ELSE
        MAKE (Door_Alarm_State = any value that differs from its initial Door_Alarm_State)
12. 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, new Status_Flags, and Door_Alarm_State)
13. TRANSMIT BACnet-SimpleACK-PDU

14. TRANSMIT SubscribeCOV-Request,
        'Subscriber Process Identifier' =     (the same value used in the SubscribeCOV-Request),
        'Monitored Object Identifier' =       X
15. RECEIVE BACnet-SimpleACK-PDU
~~16. IF (Out_Of_Service is writable) THEN~~
        ~~WRITE Out_Of_Service =         FALSE~~
```

8.2.X12 Change of Value Notification from an Access Point object

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

Purpose: To verify that the IUT can initiate ConfirmedCOVNotification service requests conveying a change of the Status_Flags and Access_Event_Time properties of Access Point 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 Access_Event_Time and Status_Flags of the monitored object *are*~~is~~ changed, and a notification shall be received. The ~~properties~~*Access_EventTime property* may be changed using the WriteProperty service or by another means. ~~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. For implementations where it is not possible to ~~write Out_Of_Service or~~ change the Status_Flags by any other means, steps 5,6,7 shall be skipped.

Configuration Requirements: At the beginning of the test, the Out_Of_Service property shall have a value of FALSE.

Test Steps:

```
1.  TRANSMIT SubscribeCOV-Request,
        'Subscriber Process Identifier' =     (PI: 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' =           PI,
            'Initiating Device Identifier' =  IUT,
            'Monitored Object Identifier' = X,
            'Time Remaining' =                (any value appropriate for the Lifetime selected),
            'List of Values' =                (the initial Access_Event, Status_Flags, Access_Event_Tag,
                                              Access_Event_Time, Access_Event_Credential and
                                              Access_Event_Authentication_Factor if X has an
                                              Access_Event_Authentication_Factor property)
4.  TRANSMIT BACnet-SimpleACK-PDU

5.  ~~IF (Out_OF_Service is writable) THEN~~
        ~~WRITE Out_OF_Service = TRUE~~
    ~~ELSE~~
        MAKE (Status_Flags = any value that differs from initial Status_Flags)
6.  BEFORE **Notification Fail Time**
        RECEIVE ConfirmedCOVNotification-Request,
            'Subscriber Process Identifier' =           PI,
            'Initiating Device Identifier' =  IUT,
            'Monitored Object Identifier' = X,
            'Time Remaining' =      (any value appropriate for the Lifetime selected),
            'List of Values' =      (the initial Access_Event, new Status_Flags, initial Access_Event_Tag,
                                    Access_Event_Time, Access_Event_Credential and
                                     Access_Event_Authentication_Factor if X has an Access_Event_Authentication_Factor
property)
7.  TRANSMIT BACnet-SimpleACK-PDU

8.  IF (Access_Event_Time is now writable) THEN
        WRITE Access_Event_Time = (any value that differs from initial Access_Event_Time)
    ELSE
        MAKE (Access_Event_Time = any value that differs from initial Access_Event_Time)
9.  BEFORE **Notification Fail Time**
      RECEIVE ConfirmedCOVNotification-Request,
        'Subscriber Process Identifier' =     PI,
        'Initiating Device Identifier' =      IUT,
        'Monitored Object Identifier' =       X,
        'Time Remaining' =          (any value appropriate for the Lifetime selected),
        'List of Values' =          (the new values of Access_Event, Access_Event_Tag, Access_Event_Time,
                                     Access_Event_Credential, and Access_Event_Authentication_Factor if X has
                                    Access_Event_Authentication_Factor property)
10. TRANSMIT BACnet-SimpleACK-PDU

11. TRANSMIT SubscribeCOV-Request,
        'Subscriber Process Identifier' =     PI,
        'Monitored Object Identifier' =       X
```

12. RECEIVE BACnet-SimpleACK-PDU
~~13. IF (Out_Of_Service is writable) THEN~~
~~WRITE Out_Of_Service = FALSE~~
~~14~~*13*. CHECK (verify that no notification message has been transmitted)

8.2.X13 Change of Value Notification from a Credential Data Input object
Reason for Change: No test exists for this functionality. This test is not in any SSPC proposal.

Purpose: To verify that the IUT can initiate ConfirmedCOVNotification service requests conveying a change of the Status_Flags and Update_Time properties of Credential Data Input 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 and Update_Time properties of the monitored object *are*~~is~~ changed, and a notification shall be received. The *Present_Value property*~~properties~~ may be changed using the WriteProperty service or by another means. ~~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. For implementations where it is not possible to ~~write Out_Of_Service or~~ change the Status_Flags by any other means, steps 5, 6, and 7 shall be skipped.

~~Configuration Requirements: At the beginning of the test, the Out_Of_Service property shall have a value of FALSE.~~

Test Steps:

1. TRANSMIT SubscribeCOV-Request,
   'Subscriber Process Identifier' = (PI: any value > 0chosen 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' = PI,
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = X,
   'Time Remaining' = (any value appropriate for the Lifetime selected),
   'List of Values' = (the initial Present_Value, initial Status_Flags, and Update_Time (most recent update time when the Present_Value was updated))
4. TRANSMIT BACnet-SimpleACK-PDU

5. ~~IF (Out_OF Service is writable) THEN~~
   ~~WRITE Out_OF Service = TRUE~~
   ~~ELSE~~
   MAKE (Status_Flags = any value that differs from initial Status_Flags)
6. BEFORE **Notification Fail Time**
   RECEIVE ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = PI,
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = X,
   'Time Remaining' = (any value appropriate for the Lifetime selected),
   'List of Values' = (the initial Present_Value, new Status_Flags, and Update_Time (most recent update time when the Present_Value was updated))
7. TRANSMIT BACnet-SimpleACK-PDU

8. IF (Present_Value is now writable) THEN
   WRITE X, Present_Value = (any value that differs from initial Present_Value)
   ELSE
   MAKE (Present_Value = any value that differs from initial Present_Value)
9. BEFORE **Notification Fail Time**

```
        RECEIVE ConfirmedCOVNotification-Request,
            'Subscriber Process Identifier' =     PI,
            'Initiating Device Identifier' =      IUT,
            'Monitored Object Identifier' =       X,
            'Time Remaining' =                    (any value appropriate for the Lifetime selected),
            'List of Values' =                    (the new Present_Value, new Status_Flags, and Update_Time
                                                  (most recent update time when the Present_Value was updated))
10.  TRANSMIT BACnet-SimpleACK-PDU

11.  Verify Update_Time received in step 7.
12.  TRANSMIT SubscribeCOV-Request,
            'Subscriber Process Identifier' =  PI,
            'Monitored Object Identifier' =       X
13.  RECEIVE BACnet-SimpleACK-PDU
~~14.      IF (Out_Of_Service is writable) THEN~~
~~                WRITE Out_Of_Service =           FALSE~~
~~15~~*14*.     CHECK (verify that no notification message has been transmitted)
```

[Modify test 9.11.1.X21 as shown]

9.11.1.X21 Confirmed Change of Value Notification from Status_Flags Property
Reason for Change: No test exists for this functionality. This test is not in any SSPC proposal.

Purpose: To verify that the IUT can initiate ConfirmedCOVNotification service requests conveying a change of the Status_Flags Property.

Test Concept: A property 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.~~ *made to change and a notification shall be received. For implementations where it is not possible to change the Status_Flags property, this test shall be skipped*.

Test Steps:

```
1. TRANSMIT SubscribeCOVProperty-Request,
        'Subscriber Process Identifier' = (any valid process identifier),
         'Monitored Object Identifier' = X
        'Issue Confirmed Notifications' = TRUE,
        'Lifetime' = L
        'Monitored Property Identifier' = Y (any valid property supporting COV notifications)
2. RECEIVE BACnet-SimpleACK-PDU
3. BEFORE Notification Fail Time
        RECEIVE BACnetConfirmedCOVNotification-Request,
            'Subscriber Process Identifier' = (the same identifier used in the subscription),
            'Initiating Device Identifier' = IUT,
            'Monitored Object Identifier' = X
            'Time Remaining' =          (any value appropriate for the Lifetime selected),
            'List of Values' =       (values appropriate to the property subscribed to and initial
 Status_Flags)
4. TRANSMIT BACnet-SimpleACK-PDU
5. MAKE (Status_Flags = any value that differs from "initial Status_Flags")
6. BEFORE Notification Fail Time
        RECEIVE BACnetConfirmedCOVNotification-Request,
            'Subscriber Process Identifier' = (the same identifier used in the subscription),
            'Initiating Device Identifier' = IUT,
            'Monitored Object Identifier' = X
            'Time Remaining' =     (any value appropriate for the Lifetime selected),
```

'List of Values' = (initial values appropriate to the property subscribed to and new Status_Flags)

7. TRANSMIT SubscribeCOVProperty-Request,
   'Subscriber Process Identifier' = (the same identifier used in Step 1),
   'Monitored Object Identifier' = X
   'Monitored Property Identifier' = Y
8. RECEIVE BACnet-SimpleACK-PDU

BTL-20.0.1 imp1-6: Update Workstation Tests to Clarify Directives [BTLWG-1187]

Overview:

In Section 6.3.4 Is able to Schedule any B-Side Device with a Protocol_Revision Less Than or Equal to its Own Protocol_Revision, the existing Testing Hints should be deleted and the new Test Directives added.

Changes:

Checklist Changes

None

Test Plan Changes

6.3 Scheduling - Weekly Schedule - A

[Modify the Scheduling section Weekly Schedule A]

6.3.4 Is able to Schedule any B-Side Device with a Protocol_Revision Less Than or Equal to its Own Protocol_Revision

The IUT is able to schedule any server device that supports schedules from the same or previous revisions of BACnet.

135.1-2019 - 13.10 - Workstation Scheduling Tests		
	Test Conditionality	Must be executed.
	Test Directives	*At least one test from 13.10.1 or 13.10.2 shall be executed against a TD with Protocol_Revision 3.*
	Testing Hints	~~A reference device with schedules supporting Protocol_Revision = 3 and a reference device with schedules supporting Protocol_Revision >= 4 should be used. Any of the tests in this section may be executed.~~

Test Changes

None

BTL-20.0.1 imp1-7: Fix the CHANGE_OF_RELIABILITY Requirements [BTLWG-1192]

Overview:

CHANGE_OF_RELIABILITY requirements in the Test Package are incorrect.

1. The test package currently allows a device to claim that CHANGE_OF_RELIABILITY as the only algorithm supported. Since this only handles ToFault, it is insufficient to claim support for alarming if this is the only EventType supported.
2. Any event generating object which is capable of having a Reliability other than NO_FAULT_DETECTED has to be able to support ToFault notifications (there is currently no testing to enfore this).
3. Any Event Enrollment object which can monitor objects outside the box must be able to generate ToFault notifications. (because the monitored object might go into fault, or communication between the devices may be lost).

Changes:

Checklist Changes

[Added new Conformance code for AE-N-I-B and AE-N-E-B BIBBS]

Alarm and Event Management - Notification - Internal - B		
	R	Base Requirements
	R	Supports AE-INFO-B
	R	Supports the Notification Class Object
	C[1]	Supports AE-ACK-B
	C[2]	Implements intrinsic alarming
	C[2]	Supports the Event Enrollment object
	C[3]	Implements the CHANGE_OF_BITSTRING algorithm
	C[3]	Implements the CHANGE_OF_STATE algorithm
	C[3]	Implements the numeric form of the CHANGE_OF_VALUE algorithm
	C[3]	Implements the Bit String form of the CHANGE_OF_VALUE algorithm
	C[3]	Implements the COMMAND_FAILURE algorithm
	C[3]	Implements the FLOATING_LIMIT algorithm
	C[3]	Implements the OUT_OF_RANGE algorithm
	C[3]	Implements a proprietary algorithm using complex notifications
	C[3]	Implements a standard or proprietary algorithm using extended notifications
	C[4]	Generates event notifications with timestamps of the BACnetDateTime form
	C[4]	Generates event notifications with timestamps of the Time form
	C[4]	Generates event notifications with timestamps of the Sequence Number form
	O	Implements intrinsic alarming in an Analog object
	O	Supports writable Event_Parameters properties
	C[3]	Implements the DOUBLE_OUT_OF_RANGE algorithm
	C[3]	Implements the SIGNED_OUT_OF_RANGE algorithm
	C[3]	Implements the UNSIGNED_OUT_OF_RANGE algorithm
	C[3]	Implements the CHANGE_OF_CHARACTERSTRING algorithm
	C[3]	Implements the CHANGE_OF_STATUS_FLAGS algorithm
	C[3]	Implements the UNSIGNED_RANGE algorithm
	O	Supports Event_Message_Texts property
	O	Supports Event_Message_Texts_Config property
	O	Implements intrinsic alarming in an Integer object
	C[3,9]	Implements the CHANGE_OF_RELIABILITY - NONE
	C[3,9]	Implements the CHANGE_OF_RELIABILITY - FAULT_CHARACTERSTRING algorithm
	C[3,9]	Implements the CHANGE_OF_RELIABILITY - FAULT_EXTENDED algorithm
	C[3,9]	Implements the CHANGE_OF_RELIABILITY - FAULT_LIFE_SAFETY algorithm
	C[3,9]	Implements the CHANGE_OF_RELIABILITY - FAULT_STATE algorithm
	C[3,9]	Implements the CHANGE_OF_RELIABILITY - FAULT_STATUS_FLAGS algorithm
	C[3,8,9]	Implements the CHANGE_OF_RELIABILITY - FAULT_LISTED algorithm
	C[3,9]	Supports CHANGE_OF_RELIABILITY in the Event Enrollment Object

	C[3,5]	Implements the CHANGE_OF_LIFE_SAFETY algorithm
	O	Implements the ACCESS_EVENT algorithm
	C[3.6]	Implements the CHANGE_OF_DISCRETE_VALUE algorithm
	C[3,7]	Implements the CHANGE_OF_TIMER algorithm
	C[3,6,9]	Implements the CHANGE_OF_RELIABILITY - FAULT_OUT_OF_RANGE algorithm
	O	Implements the Reliability_Evaluation_Inhibit property
[1] Required if EventNotifications with service parameter AckRequired = True can be issued. [2] At least one of these options must be supported to claim support for this BIBB. [3] At least one of these options must be supported to claim support for this BIBB. It is recommended that a standard BACnet algorithm be used instead of a proprietary algorithm whenever possible. [4] At least one of these options must be supported to claim support for this BIBB. The BACnetDateTime form of the timestamp is the recommended option. [5] Contact BTL for interim tests for this algorithm. [6] Protocol_Revision 16 or higher must be claimed. [7] Protocol_Revision 17 or higher must be claimed. [8] Protocol_Revision 18 or higher must be claimed. [9] A device shall support CHANGE_OF_RELIABILITY in any object which generates event notifications and in which the Reliability property can take on a value other than NO_FAULT_DETECTED.		
Alarm and Event Management - Notification - External - B		
	R	Base Requirements
	R	Supports AE-N-I-B
	R	Supports DS-RP-A for retrieving monitored values
	R	Supports the Event Enrollment object
	C[1]	Implements the CHANGE_OF_BITSTRING algorithm
	C[1]	Implements the CHANGE_OF_STATE algorithm
	C[1]	Implements the numeric form of the CHANGE_OF_VALUE algorithm
	C[1]	Implements the Bit String form of the CHANGE_OF_VALUE algorithm
	C[1]	Implements the COMMAND_FAILURE algorithm
	C[1]	Implements the FLOATING_LIMIT algorithm
	C[1]	Implements the OUT_OF_RANGE algorithm
	C[1]	Implements the DOUBLE_OUT_OF_RANGE algorithm
	C[1]	Implements the SIGNED_OUT_OF_RANGE algorithm
	C[1]	Implements the UNSIGNED_OUT_OF_RANGE algorithm
	C[1]	Implements the CHANGE_OF_CHARACTERSTRING algorithm
	C[1]	Implements the CHANGE_OF_STATUS_FLAGS algorithm
	C[1]	Implements the UNSIGNED_RANGE algorithm
	C[1,6]	Implements the CHANGE_OF_RELIABILITY - NONE
	C[1,6]	Implements the CHANGE_OF_RELIABILITY - FAULT_CHARACTERSTRING algorithm
	C[1,6]	Implements the CHANGE_OF_RELIABILITY - FAULT_EXTENDED algorithm
	C[1,6]	Implements the CHANGE_OF_RELIABILITY - FAULT_LIFE_SAFETY algorithm
	C[1,6]	Implements the CHANGE_OF_RELIABILITY - FAULT_STATE algorithm
	C[1,6]	Implements the CHANGE_OF_RELIABILITY - FAULT_STATUS_FLAGS algorithm
	C[1,5,6]	Implements the CHANGE_OF_RELIABILITY - FAULT_LISTED algorithm
	C[1,6]	Supports CHANGE_OF_RELIABILITY in the Event Enrollment object
	C[1,2]	Implements the CHANGE_OF_LIFE_SAFETY algorithm
	O	Implements the ACCESS_EVENT algorithm
	C[1,3]	Implements the CHANGE_OF_DISCRETE_VALUE algorithm
	C[1,4]	Implements the CHANGE_OF_TIMER algorithm
	C[1,3,6]	Implements the CHANGE_OF_RELIABILITY - FAULT_OUT_OF_RANGE algorithm
	C[1]	Implements a Proprietary algorithm
	O	Supports Event_Message_Texts property

[1] At least one of these options must be supported to claim support for this BIBB. It is recommended that a standard BACnet algorithm be used instead of a proprietary algorithm whenever possible.
[2] Contact BTL for interim tests for this algorithm.
[3] Protocol_Revision 16 or higher must be claimed.
[4] Protocol_Revision 17 or higher must be claimed.
[5] Protocol_Revision 18 or higher must be claimed.
[6] A device shall support CHANGE_OF_RELIABILITY in any object which generates event notifications and in which the Reliability property can take on a value other than NO_FAULT_DETECTED.

Test Plan Changes

5.2.37 Supports CHANGE_OF_RELIABILITY in the Event Enrollment Object

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

BTL - 8.5.17.9 - CHANGE_OF_RELIABILITY of Event Enrollment Object Fault (UnconfirmedEventNotifications)		
	Test Conditionality	~~If the IUT has no Event Enrollment object that detects an internal unreliable operational fault, this test shall be skipped.~~ *Must be executed.*
	Test Directives	
	Testing Hints	

5.3.25 Supports CHANGE_OF_RELIABILITY in the Event Enrollment Object

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

BTL - 8.5.17.7.1 - Internal Faults Take Precedence Over Monitored Object Faults		
	Test Conditionality	~~If the IUT does not support an Event Enrollment object which can detect internal faults and monitor an object which detects faults, then this test shall be skipped.~~ *Must be executed.*
	Test Directives	*Execute test using an Event Enrollment object monitoring an object (O1) in a device other than the IUT.*
	Testing Hints	
BTL - 8.5.17.7.2 - Monitored Object Faults Take Precedence Over Fault Algorithms		
	Test Conditionality	If the IUT does not support an Event Enrollment object which monitors an object which detects faults and which applies a fault algorithm, then this test shall be skipped.
	Test Directives	*Execute test using an Event Enrollment object monitoring an object (O1) in a device other than the IUT.*
	Testing Hints	
BTL - 8.5.17.7.3 - Internal Faults Take Precedence Over Fault Algorithms		
	Test Conditionality	If the IUT does not support an Event Enrollment object which can detect internal faults and which applies a fault algorithm, then this test shall be skipped.
	Test Directives	*Execute test using an Event Enrollment object monitoring an object (O1) in a device other than the IUT.*
	Testing Hints	
BTL - 8.5.17.8 - CHANGE_OF_RELIABILITY of Event Enrollment Object, Monitored Object Fault (UnconfirmedEventNotifications)		
	Test Conditionality	~~If the IUT has no Event Enrollment object where the Monitored_Object that can transition to fault, this test shall be skipped.~~ *Must be executed.*
	Test Directives	*Execute test using an Event Enrollment object monitoring an object (M1) in a device other than the IUT.*

	Testing Hints	
BTL - 8.5.17.9 - CHANGE_OF_RELIABILITY of Event Enrollment Object Fault (UnconfirmedEventNotifications)		
	Test Conditionality	~~If the IUT has no Event Enrollment object that detects an internal unreliable operational fault, this test shall be skipped.~~ *Must be executed.*
	Test Directives	*Execute test using an Event Enrollment object monitoring an object (M1) in a device other than the IUT.*
	Testing Hints	

Test Changes

None

BTL-20.0.1 imp1-8: Add Checklist Hint [BTLWG-1240]

Overview:

While individual BTL Checklist sections contain a footnote indicating that the applicant shall contact BTL for interim tests, a generic note on that matter is missing.

Changes:

Checklist Changes

[In BTL Checklist, modify Clause 1.1 - IUT Information]

This section shall provide additional information about the IUT when the BTL Checklist is filled out.

Fill out date
Vendor Name
Product Name
Product Model Number
BACnet Protocol Version 1
BACnet Protocol Revision [1]
BACnet Device Profile
BACnet Firmware Revision
BACnet Application Software Version

Remarks

[1] *If the IUT claims a Protocol_Revision higher than 23, contact the BTL for interim tests for the product.*

Test Plan Changes

None

Test Changes

None

BTL-20.0.1 imp1-9: Add Directives for File Tests [BTLWG-1279]

Overview:

The tests in section **3.61.4 Supports a Stream-Based File Object for a Purpose Other Than Backup and Restore** need directives indicating that the tests must be applied to a file which is not used for Backup and Restore.

Changes:

Checklist Changes

None

Test Plan Changes

[Modify Test Plan Section 3.61.4 and 3.61.5, adding the below directives to each entry, except test 135.2019 - 9.12.2.2.2 - Attempting to Read Data from a Nonexistent file.]

	Test Directives	*Apply to a file not used for Backup & Restore.*

Test Changes

None