BACnet® TESTING LABORATORIES ADDENDA

Addendum cr2 to BTL Test Package 23.3

Revision final
Revised 10/7/2024

Approved by the BTL Working Group on October 3, 2024;
Approved by the BTL Working Group Voting Members on October 17, 2024;
Published on October 22, 2024.

[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 23.3 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-23.3 cr2-1: Scheduling Internal B Should not Require Multiple Special Events [BTLWG-1102, CR-0486]

Overview:

The test package currently requires that devices support multiple special events. CR-0486 pointed this out and the BTL-WG's response was:

> Any test that requires more than one special event or more than one time-value pair may be skipped if the IUT does not support such.

The test conditionality for any test under SCHED-I-B which does require this should be changed.

Changes:

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, modify the test conditionality for the following tests as shown.]

Scheduling - Internal - B

6.4.1 Base Requirements

Base requirements must be met by any IUT claiming conformance to this BIBB. (The BIBB requires, among other things, support for either TimeSynchronization-Request or UTCTimeSynchronization-Request execution; these are tested by the Device Management tests.)

…		
135.1-2023 - 7.3.2.23.3.8 - Event Priority Test		
	Test Conditionality	~~This test shall be executed if and only if the IUT is prior to protocol revision 4. If the IUT is of the correct Protocol_Revision, the IUT is required to be configurable such that this test can be run. This test may not be skipped.~~ If the IUT claims protocol revision 4 or higher, this test shall be skipped. If the IUT does not support more than one exception schedule entry, the test shall be skipped.
	Test Directives	
	Testing Hints	
BTL - 7.3.2.23.10.3.8 - Revision 4 Event Priority Test		
	Test Conditionality	~~If the IUT does not support enough exception schedule entries to execute this test, the test shall be skipped, otherwise the test shall be executed.~~ If the IUT claims protocol revision 3 or prior, this test shall be skipped. If the IUT does not support more than one exception schedule entry, the test shall be skipped.
	Test Directives	
	Testing Hints	
135.1-2023 - 7.3.2.23.10.3.10 - Revision 4 Calendar Entry WeekNDay Odd-Numbered Month Test		
	Test Conditionality	~~This test shall be executed if and only if the IUT is protocol revision 4 or higher. If the IUT is of the correct Protocol_Revision, the IUT is required to be configurable such that this test can be run. This test may not be skipped.~~ If the IUT claims protocol revision 3 or prior, this test shall be skipped.

		If the IUT does not support more than one exception schedule entry, the test shall be skipped.
	Test Directives	
	Testing Hints	
135.1-2023 - 7.3.2.23.10.3.11 - Revision 4 Calendar Entry WeekNDay Even-Numbered Month Test		
	Test Conditionality	~~This test shall be executed if and only if the IUT is protocol revision 4 or higher. If the IUT is of the correct Protocol_Revision, the IUT is required to be configurable such that this test can be run. This test may not be skipped.~~ If the IUT claims protocol revision 3 or prior, this test shall be skipped. If the IUT does not support more than one exception schedule entry, the test shall be skipped.
	Test Directives	
	Testing Hints	
135.1-2023 - 7.3.2.23.10.3.12 - Revision 4 Lower Event Priority Change Test		
	Test Conditionality	~~If the IUT does not support enough exception schedule entries~~ If the IUT does not support more than one exception schedule entry, or not more than 2 time-value pair in exception schedule, the test shall be skipped.
	Test Directives	
	Testing Hints	
…		

Specified Test Changes

None

BTL-23.3 cr2-2: Resolve Duplicate Connection Test [BTLWG-1290, CR-0528]

Overview:

The test mandates a specific order of connect and disconnect requests when duplicate connection requests occur. See CR-0528.

Changes:

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, change all test references of 135.1-2023 - 12.5.2.1.9 to BTL]

9.9.3 Is Able to Operate as a Hub

The IUT is able to operate as a hub (contains a hub function).

…		
BTL - 12.5.2.1.9 - Duplicate Connection Test		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
…		

Specified Test Changes

12.5.2.1.9 Duplicate Connection Test

Reason for Change: The test mandates a specific order of connect and disconnect requests when duplicate connection requests occur. See CRR-0528.

Purpose: To verify that duplicate hub connection requests result in the original connection being dropped.

Test Concept: With the IUT operating as hub, connect device D3 to the IUT's hub URI. *While maintaining this first connection*~~When the connection is complete~~, *make* D3 *establish* ~~attempts to bring up~~ a second connection to the IUTs hub URI *with the same VMAC*. Verify that the IUT accepts the second connect request and closes the first connection. *Then, while maintaining the second connection, make D3 establish another new connection (third)*~~Repeat the reconnection, but~~ with a new VMAC for *D3;*~~D3 and~~ ensure that the new *connect-*request is accepted and the ~~existing~~ *second connection is closed*~~one dropped~~.

Notes to Tester: For Steps 6, 7, and 8 and Steps 12, 13, and 14, the order in which the IUT transitions from the existing connection to the new connection is not significant.

Test Steps:

1. MAKE(D3 connect to the IUT's hub function)
2. TRANSMIT PORT (D3-IUT hub first WebSocket),
   Connect-Request,
   'Message ID' = (M1: any valid value),
   -- 'Originating Virtual Address' absent
   -- 'Destination Virtual Address' absent

-- 'Destination Options' absent
-- 'Data Options' absent
'VMAC Address' = (D3's VMAC),
'Device UUID' = (D3's UUID),
'Maximum BVLC Length' = (the D3's maximum BVLC accepted length),
'Maximum NPDU Length' = (the D3's maximum NPDU accepted length)

3. RECEIVE PORT (D3-IUT hub first WebSocket),
   Connect-Accept,
   'Message ID' = M1,
   -- 'Originating Virtual Address' absent
   -- 'Destination Virtual Address' absent
   'VMAC Address' = (IUT's VMAC),
   'Device UUID' = (IUT's UUID),
   'Maximum BVLC Length' = (the IUT's maximum BVLC accepted length),
   'Maximum NPDU Length' = (the IUT's maximum NPDU accepted length)
4. MAKE(~~D1~~*D3* connect a second WebSocket to the IUT's hub function)
5. TRANSMIT PORT (D3-IUT hub second WebSocket),
   Connect-Request,
   'Message ID' = (M2: any valid value),
   -- 'Originating Virtual Address' absent
   -- 'Destination Virtual Address' absent
   -- 'Destination Options' absent
   -- 'Data Options' absent
   'VMAC Address' = (D3's VMAC),
   'Device UUID' = (D3's UUID),
   'Maximum BVLC Length' = (the D3's maximum BVLC accepted length),
   'Maximum NPDU Length' = (the D3's maximum NPDU accepted length)
6. RECEIVE PORT (D3-IUT hub second WebSocket),
   Connect-Accept,
   'Message ID' = M2,
   -- 'Originating Virtual Address' absent
   -- 'Destination Virtual Address' absent
   'Destination Options' = (absent or a list of valid options),
   -- 'Data Options' absent
   'VMAC Address' = (IUT's VMAC),
   'Device UUID' = (IUT's UUID),
   'Maximum BVLC Length' = (the IUT's maximum BVLC accepted length),
   'Maximum NPDU Length' = (the IUT's maximum NPDU accepted length)
7. RECEIVE PORT (D3-IUT hub first WebSocket),
   Disconnect-Request,
   'Message ID' = M3,
   -- 'Originating Virtual Address' absent
   -- 'Destination Virtual Address' absent
   'Destination Options' = (absent or a list of valid options),
   -- 'Data Options' absent
8. TRANSMIT PORT (D3-IUT hub first WebSocket),
   Disconnect-ACK,
   'Message ID' = M3,
   -- 'Originating Virtual Address' absent
   -- 'Destination Virtual Address' absent
   -- 'Destination Options' absent
   -- 'Data Options' absent
9. CHECK(that the IUT closed D3's initial WebSocket)
10. MAKE(D3 connect a third WebSocket to the IUT's hub function)
11. TRANSMIT PORT (D3-IUT hub second WebSocket),
    Connect-Request,
    'Message ID' = (M4: any valid value),
    -- 'Originating Virtual Address' absent
    -- 'Destination Virtual Address' absent
    -- 'Destination Options' absent

-- 'Data Options' absent
'VMAC Address' = (a new VMAC for D3 which does not conflict with any other VMACs),
'Device UUID' = (D3's UUID),
'Maximum BVLC Length' = (the D3's maximum BVLC accepted length),
'Maximum NPDU Length' = (the D3's maximum NPDU accepted length)

12. RECEIVE PORT (D3-IUT hub third WebSocket),
Connect-Accept,
'Message ID' = M4,
-- 'Originating Virtual Address' absent
-- 'Destination Virtual Address' absent
'Destination Options' = (absent or a list of valid options),
-- 'Data Options' absent
'VMAC Address' = (IUT's VMAC),
'Device UUID' = (IUT's UUID),
'Maximum BVLC Length' = (the IUT's maximum BVLC accepted length),
'Maximum NPDU Length' = (the IUT's maximum NPDU accepted length)
13. RECEIVE PORT (D3-IUT hub second WebSocket),
Disconnect-Request,
'Message ID' = M5,
-- 'Originating Virtual Address' absent
-- 'Destination Virtual Address' absent
'Destination Options' = (absent or a list of valid options),
-- 'Data Options' absent
14. TRANSMIT PORT (D3-IUT hub second WebSocket),
Disconnect-ACK,
'Message ID' = M5,
-- 'Originating Virtual Address' absent
-- 'Destination Virtual Address' absent
-- 'Destination Options' absent
-- 'Data Options' absent
15. CHECK(that the IUT closed D3's second WebSocket)

BTL-23.3 cr2-3: Add Additional Test Directives for Test 8.22.5 in DS-LSM-A [BTLWG-1317, CR-0536]

Overview:

As per test plan, 8.22.5 Accepting Input and Commanding or Relinquishing Properties test conditionality is "must be executed" against commendable properties. However, both the life safety point and the life safety zone object do not support any commendable properties. In my opinion, this test case will be skipped against life-safe objects and will be executed only against the BACnet clause of 19.2.1.1 of 135-2020 mentioned properties.

Question:

1. Is my interpretation correct? >> **Yes**
2. If yes, Can BTL add a note or appropriate statement in the test conditionality section or in the test case for the same? >> **Yes**

Changes:

Checklist Changes

None

Test Plan Changes

[Add additional Test Directives for test 8.22.5 section 4.29 Data Sharing - Life Safety Modify - A, p.273]

4.29 Data Sharing - Life Safety Modify - A

4.29.1 Base Requirements

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

135.1-2023 - 8.22.4 - Accepting Input and Modifying Properties		
	Test Conditionality	Must be executed if the IUT does not support DS-LSAM-A.
	Test Directives	Repeat the test for each of the required object types listed in the BIBB definition. Repeat for each of the required properties listed in the BIBB definition, except for those properties which are commandable. Repeat the test for a variety of values that cover the range of values required by the "Minimum Writable Value Ranges" table in the DS-M-A BIBB definition.
	Testing Hints	
135.1-2023 - 8.22.5 - Accepting Input and Commanding/Relinquishing Properties		
	Test Conditionality	Must be executed if the IUT does not support DS-LSAM-A.
	Test Directives	Repeat the test for all commandable properties defined for the claimed Protocol_Revision. This test should be executed at priority 8 only, i.e. $PR_1 = 8$.
	Testing Hints	

Specified Test Changes

None

BTL-23.3 cr2-4: Changes to Accommodate pTimeDelay=0 [BTLWG-1472, CR-0561]

Overview:

In cases where the IUT's pTimeDelay is 0, there is not enough time to execute the VERIFY steps in test 8.5.17.10. Change allows VERIFY steps to be skipped in these cases.

Changes:

Checklist Changes

[None]

Test Plan Changes

[In BTL Test Plan, change all references to test 8.5.17.10 from 135.1-2023 to BTL. Currently 2.]

Specified Test Changes

[Add test 8.5.17.10 to BTL Specified Tests from 135.1-2023]

8.5.17.10 After FAULT-to-NORMAL, Re-Notification of OFFNORMAL (UnconfirmedEventNotifications)

Reason for Change: Account for cases where pTimeDelay=0.

Purpose: To verify that objects go to the NORMAL state after leaving the FAULT state, then transition to OFFNORMAL if the condition still exists.

Test Concept: Select a fault detecting object, O1, which is able to detect OFFNORMAL conditions. Make O1 transition to an OFFNORMAL state and then transition to FAULT. Remove the condition causing the FAULT and verify O1 transitions from FAULT to NORMAL, then verify that the object transitions from NORMAL to the original OFFNORMAL state.

Configuration Requirements: O1 is configured to detect and report faults. O1 is configured to have no fault conditions present, and Event_State is NORMAL. *If writable or configurable, set the value of the Time_Delay property in O1 to a value greater than zero*. The 'Issue Confirmed Notifications' parameter shall have a value of FALSE.

Test Steps:

1. VERIFY pCurrentReliability = NO_FAULT_DETECTED
2. VERIFY pCurrentState = NORMAL
3. MAKE (O1transition to an offnormal state)
4. WAIT pTimeDelay
5. BEFORE **Notification Fail Time**
6. RECEIVE UnconfirmedEventNotification-Request
   'Process Identifier' = (any valid process identifier),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = O1,
   'Time Stamp' = (any valid time stamp),
   'Notification Class' = (the notification class configured for O1),
   'Priority' = (the value configured for the transition),
   'Event Type' = (ET1, any valid off normal event type),
   'Message Text' = (optional, any valid message text),
   'Notify Type' = ALARM | EVENT,
   'AckRequired' = TRUE | FALSE,
   'From State' = NORMAL,

'To State' = OFFNORMAL,
'Event Values' = (property-values appropriate for O1)

7. VERIFY pCurrentState = OFFNORMAL
8. MAKE (O1 enter a fault state)
9. BEFORE **Notification Fail Time**
10. RECEIVE UnconfirmedEventNotification-Request
'Process Identifier' = (any valid process identifier),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = O1,
'Time Stamp' = (any valid time stamp),
'Notification Class' = (the notification class configured for O1),
'Priority' = (the value configured for the transition),
'Event Type' = CHANGE_OF_RELIABILITY,
'Message Text' = (optional, any valid message text),
'Notify Type' = ALARM | EVENT,
'AckRequired' = TRUE | FALSE,
'From State' = OFFNORMAL,
'To State' = FAULT,
'Event Values' = ((R1 any valid BACnetReliability),
(?, T, ?, ?),
(A list of valid values for properties required to be reported for O1, and 0 or more other properties of O1))
11. MAKE (O1 clear the fault condition)
12. BEFORE **Notification Fail Time**
13. RECEIVE UnconfirmedEventNotification-Request
'Process Identifier' = (any valid process identifier),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = O1,
'Time Stamp' = (TS1, any valid time stamp),
'Notification Class' = (the notification class configured for O1),
'Priority' = (the value configured for the transition),
'Event Type' = CHANGE_OF_RELIABILITY,
'Message Text' = (optional, any valid message text),
'Notify Type' = ALARM | EVENT,
'AckRequired' = TRUE | FALSE,
'From State' = FAULT,
'To State' = NORMAL,
'Event Values' = (NO_FAULT_DETECTED,
(F, F, ?, ?),
(A list of valid values for properties required to be reported for O1, and 0 or more other properties of O1))
14. *IF (pTimeDelay > 0) THEN {*
15. VERIFY pCurrentReliability = NO_FAULT_DETECTED
16. VERIFY pCurrentState = NORMAL
17. WAIT ~~until TS1 +~~ pTimeDelay
}
18. BEFORE **Notification Fail Time**
19. RECEIVE UnconfirmedEventNotification-Request
'Process Identifier' = (any valid process identifier),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = O1,
'Time Stamp' = (any valid time stamp),
'Notification Class' = (the notification class configured for O1),
'Priority' = (the value configured for the transition),
'Event Type' = ET1,
'Message Text' = (optional, any valid message text),
'Notify Type' = ALARM | EVENT,
'AckRequired' = TRUE | FALSE,
'From State' = NORMAL,
'To State' = OFFNORMAL,

'Event Values' = (property-values appropriate for O1)
20. VERIFY pCurrentReliability = NO_FAULT_DETECTED
21. VERIFY pCurrentState = OFFNORMAL

BTL-23.3 cr2-5: Add missing test 8.4.17.10 to Test Plan Section AE-N-I-B [BTLWG-1535, CR-0560]

Overview:

Based on clarification request BTL-CR-0560, test 8.4.17.10 needs to be added to Test Plan section 5.2.1

Changes:

Checklist Changes

None

Test Plan Changes

[In the Test Plan section, deletions should be shown in ~~strikethrough~~, and additions in *italics*]
[If a complete new section is being added in, do not use italics]

[Modify the Base Requirements section for AE-N-I-B]

5.2.1 Base Requirements

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

[Tests without changes omitted]

135.1-2023 - 8.5.17.10 - After FAULT-to-NORMAL, Re-Notification of OFFNORMAL (UnconfirmedEventNotifications)		
	Test Conditionality	If the IUT has no object in which CHANGE_OF_RELIABILITY is implemented in an object that can be configured into an offnormal state, this test shall be skipped.
	Test Directives	The objects 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 shall be selected.
	Testing Hints	
135.1-2023 - 8.4.17.10 - After FAULT-to-NORMAL, Re-Notification of OFFNORMAL (ConfirmedEventNotifications)		
	Test Conditionality	If the IUT has no object in which CHANGE_OF_RELIABILITY is implemented in an object that can be configured into an offnormal state, this test shall be skipped.
	Test Directives	The objects 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 shall be selected.
	Testing Hints	

Specified Test Changes

None

BTL-23.3 cr2-6: Alarm and Event Requirements for Lighting, Access Control and Elevator Objects [BTLWG-1508, CR-0565]

Overview:

Based on clarification request BTL-CR-0565.

Changes:

Checklist Changes

None

Test Plan Changes

5.19.4 Supports DM-OCD-A

The IUT shall support DS-OCD-A in order to facilitate creation and deletion of~~allow the user to create~~ Event Enrollment, ~~and~~ Notification Class, and Notification Forwarder objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DM-OCD-A and that it claims the ability to create and delete Event Enrollment, Notification Class, and Notification Forwarder objects.
	Testing Hints	

5.30.4 Supports DM-OCD-A

The IUT shall support DM-OCD-A in order to facilitate creation and deletion of ~~life safety~~ Event Enrollment, Notification Class, and Notification Forwarder objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DM-OCD-A and that all object types required by AE-LSAVM-A are claimed within DM-OCD-A.
	Testing Hints	

5.35.4 Supports DM-OCD-A

The IUT shall support DM-OCD-A in order to facilitate creation and deletion of Event Enrollment, Notification Class, and Notification Forwarder ~~life safety~~ objects.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DM-OCD-A and that all object types required by AE-ACAVM-A~~DS-ACAVM-A~~ are claimed within DM-OCD-A.
	Testing Hints	

5.39.4 Supports DM-OCD-A

The IUT shall support DM-OCD-A in order to facilitate creation and deletion of Event Enrollment, Notification Class, and Notification Forwarder ~~life safety~~ objects.

Verify Checklist		
	Test Conditionality	Must be executed.

	Test Directives	Verify that the IUT claims support for DM-OCD-A and that all object types required by AE-EAVM-A are claimed within DM-OCD-A~~AE-OCD-A~~.
	Testing Hints	

8.33.2 Supports DM-OCD-A

The IUT shall support DM-OCD-A in order to create and delete ~~Access Control~~ objects as required by DS-LOM-A.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for DM-OCD-A, and that all object types required by DS-LOM-A are claimed within DM-OCD-A.
	Testing Hints	

Specified Test Changes

None

BTL-23.3 cr2-7: Attempting to Write When No Records Exist [BTLWG-1203, CR-0512]

Overview:

Step 21 does not check if a record exists before attempting to write the record.

Changes:

Checklist Changes

None

Test Plan Changes

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.

~~135.1-2023~~BTL - 13.8.1.1 - Execution of Full Backup and Restore Procedure		
	Test Conditionality	Must be executed.
	Test Directives	
	Testing Hints	
…		

Specified Test Changes

[Change 13.8.1.1]

13.8.1.1 Execution of Full Backup and Restore Procedure

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.

[NOTE: Test Steps have been updated throughout this test. They are not change marked for clarity. Also, pay close attention to the new open and close braces i.e. {} throughout the test.]
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 {
5.      READ NAMES[X] = (OL1[X]), Object_Name
    }
6.  IF (Protocol_Revision is present AND Protocol_Revision >= 10) THEN {
7.      IF (Backup_Preparation_Time is present) THEN
8.          READ BPT = Backup_Preparation_Time
        ELSE
9.          READ BPT = APDU_Timeout
10.     IF (Restore_Preparation_Time is present) THEN
11.         READ RPT = Restore_Preparation_Time
        ELSE
12.         READ RPT = APDU_Timeout
13.     IF (Restore_Completion_Time is present) THEN
14.         READ RCT = Restore_Completion_Time
        ELSE
15.         READ RCT = APDU_Timeout
16.     IF (Backup_And_Restore_State is present or Protocol_Revision >= 13) THEN
17.         VERIFY Backup_And_Restore_State = IDLE
    }
18. TRANSMIT ReinitializeDevice-Request,
        'Reinitialized State of Device' =      STARTBACKUP,
        'Password' = (any valid password)
19. RECEIVE BACnet-SimpleACK-PDU
20. IF (Protocol_Revision is present AND Protocol_Revision >= 10) THEN {
21.     WAIT BPT
22.     IF (Backup_And_Restore_State is present or Protocol_Revision >= 13) THEN {
23.         READ BRSTATE = Backup_And_Restore_State
24.         READ CF = Configuration_Files
25.         WHILE (BRSTATE = PREPARING_FOR_BACKUP) DO {
26.             WAIT 1 second
27.             READ BRSTATE = Backup_And_Restore_State
28.             IF CF is an empty list THEN
29.                 READ CF = Configuration_Files
30.             IF CF is a non-empty list THEN
31.                 READ X = (the file referenced by Configuration_Files[1]).Name
            }
32.         CHECK (BRSTATE = PERFORMING_A_BACKUP)
        }
    }
33. READ CF = Configuration_Files
34. CHECK (CF is a non-empty array of BACnetObjectIdentifiers referring to File objects)
35. REPEAT X = (each entry in CF) DO {
36.     READ Y = X, File_Access_Method
37.     IF (Y = RECORD_ACCESS) THEN {
38.         WHILE (the last read resulted in an Ack with 'End Of File' == FALSE) DO {
39.             TRANSMIT AtomicReadFile-Request,
                    'Object Identifier' = X,
                    'File Start Record' = (the next unread record),
                    'Requested Record Count' = 1
40.             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 {
41.         WHILE (the last read did not indicate 'End Of File') DO {
42.             TRANSMIT AtomicReadFile-Request,
                    'Object Identifier' = X,
                    'File Start Position' = (the next unread octet),
                    'Requested Octet Count' = MROC
43.             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 if 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
            }
        }
    }
44. TRANSMIT ReinitializeDevice-Request,
        'Reinitialize State Of Device' = ENDBACKUP,
        'Password' = (any valid password)
45. RECEIVE BACnet-SimpleACK-PDU
46. VERIFY System_Status ! = BACKUP_IN_PROGRESS
47. IF (Backup_And_Restore_State is present or (Protocol_Revision is present and Protocol_Revision >= 13)) THEN
48.     VERIFY Backup_And_Restore_State = IDLE
49. IF (Database_Revision is changeable) THEN {
50.     MAKE (the configuration in the IUT different, such that the Database_Revision property increments)
51.     VERIFY Database_Revision <> DR1
52.     READ DR2 = Database_Revision
53.     CHECK (DR1 <> DR2)
    }
54. TRANSMIT ReinitializeDevice-Request,
        'Reinitialize State Of Device' = STARTRESTORE,
        'Password' = (any valid password)
55. RECEIVE BACnet-SimpleACK-PDU
56. IF (Protocol_Revision is present AND Protocol_Revision >= 10) THEN
57.     WAIT RPT
58.     IF (Backup_And_Restore_State is present or Protocol_Revision >= 13) THEN {
59.         READ BRSTATE = Backup_And_Restore_State
60.         WHILE (BRSTATE = PREPARING_FOR_RESTORE) DO {
61.             WAIT 1 second
62.             READ BRSTATE = Backup_And_Restore_State
            }
63.         CHECK (BRSTATE = PERFORMING_A_RESTORE)
        }
64. READ OL2 = Object_List
65. REPEAT X = (entry in CF) DO {
66.     BU_FS = (the file size of X)
67.     IF (X is not in OL2) THEN {
68.         TRANSMIT CreateObject-Request
                'Object Specifier' = X
69.         RECEIVE CreateObject-ACK
                'Object Identifier' = X
        }
        READ FS = X, File_Size
70.     IF (X, File_Size <> BU_FSis not equal to the size of the backed up file) THEN
71.         WRITE X, File_Size = 0
72.     IF (Y = RECORD_ACCESS) THEN {
73.         IF (BU_FS > 0) THEN {
74.             TRANSMIT AtomicWriteFile-Request
                    'File Identifier' = X
```

```
                    'File Start Record' = 0
                    'Record Data' = (file content for first record obtained in step 11)
75.             RECEIVE AtomicWriteFile-ACK
                    'File Start Record' = 0
76.             REPEAT REC = (each record in the backup of this file) {
77.                 TRANSMIT AtomicWriteFile-Request
                        'File Identifier' = X
                        'File Start Record' = -1
                        'Record Count' = 1
                        'Record Data' = REC
78.                 RECEIVE AtomicWriteFile-ACK
                        'File Start Record' = -1
                }
            }
        }
        ELSE {
79.         REPEAT Z = (0 through the file size, in increments of MWDL) DO {
80.             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)
81.             RECEIVE AtomicWriteFile-ACK
                    'File Start Position' = Z
            }
        }
    }
82. IF (Backup_And_Restore_State is present or (Protocol_Revision is present and Protocol_Revision >= 13)) THEN
83.     VERIFY Backup_And_Restore_State = PERFORMING_A_RESTORE
84. TRANSMIT ReinitializeDevice-Request,
        'Reinitialize State Of Device' = ENDRESTORE,
        'Password' = (any valid password)
85.. RECEIVE BACnet-SimpleACK-PDU
86. IF (Protocol_Revision is present AND Protocol_Revision >= 10) THEN {
87.     WAIT RCT
88.     IF (Backup_And_Restore_State is present or Protocol_Revision >= 13) THEN
89.         VERIFY Backup_And_Restore_State = IDLE
    }
90. READ DR3 = Database_Revision
91. CHECK (DR3 <> DR1)
92. IF (Database_Revision was changed in step 16) THEN
93      CHECK (DR3 <> DR2)
94. VERIFY Last_Restore_Time > LRT1
95. READ OL3 = Object_List
96. CHECK (that OL1 and OL3 contain the same set of objects)
97. REPEAT X = (1 through length of OL1) DO {
97.     VERIFY (OL1[X]), Object_Name = NAMES[X]
    }
```