

Clarification Request

References: Specified Tests 18.1.Final - 13.8.1.1

Date of BTL-WG Response: 2021-12-09

Background:

- **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 \geq 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 \geq 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) THEN
        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) THEN
        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) THEN
            WRITE X, File_Size = 0
            IF (size of the backed up file is greater than zero) THEN
                IF (Y = RECORD_ACCESS) THEN
                    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

```

