

Clarification Request

References: ASHRAE 135.1-2025

Date of BTL-WG Response: July 23, 2026

Background:

ASHRAE 135.1-2025

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.

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
}
70. IF (X, File_Size <> BU_FS) 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 {
98. VERIFY (OL1[X]), Object_Name = NAMES[X]
}
}

```

Problem:

1. The test should run regardless of File_Access_Method (RECORD_ACCESS or STREAM_ACCESS).

In Steps 70-71 the file size should be read and written. That is not possible in case of File_Access_Method = RECORD_ACCESS, because then the File_Size is not writable.

2. According to 135-2024, Chapter 12.13.6 File_Size:

"If the size of the file is unknown, an attempt to read this property shall result in an 'Error Class' of PROPERTY and an 'Error Code' of UNKNOWN_FILE_SIZE."

Question:

1. Is it right, that the tests runs regardless of the File_Access_Method? What should the test do, if File_Access_Method = RECORD_ACCESS?

Should the test maybe read the File_Access_Method?

2. What should the test do, if File_size = UNKNOWN_FILE_SIZE ?

Response:

1. **Yes, the DM-BR-B BIBB requires the File_Size property to be writable if the size of the configuration file can change based on the device's configuration. This requirement is independent of whether the File_Access_Method is RECORD_ACCESS or STREAM_ACCESS. See 135-2024, K.5.18 and 19.1.3.3 and Interpretation Request IC 135-2012-13.**
2. **The restore procedure in 19.1.3.3 does not specify what to do in this case. An Interpretation Request will be submitted to SSPC 135.**