

Clarification Request

References: Test Plan 12.0.final

Date of BTL-WG Response: September 13, 2012

Background:

BTL test plan 12

A number of tests in testplan 12.0 final reference Addendum 135.1-2009m. This document does not appear in the list of referenced documents though. More important this addendum is still in development und not published at all!

See e.G.

135.1-2009m-4 - 13.X.2.1 - Initiate a Full Backup and Restore
--

Question:

Should all of these tests currently be ignored in testlabs and in pretesting by vendors?
(This is a proper Yes/No question :-) ?)

Response:

Given that the addendum was not published as quickly as expected, we have extracted the tests from the addendum as published and present them here.

Note, the Addendum in question was published against 135.1-2011 not 135.1-2009.

135.1-2011m-4. Add Backup and Restore Tests.

Rationale

This section adds tests for the Backup and Restore procedure. There are currently no tests for this functionality.

[Add new **Clause 13.8**, p. 516]

13.8 Backup and Restore Procedure Tests

13.8.1 Backup and Restore Execution Tests

The tests in this section verify that a device that can be backed up and restored has implemented the Backup and Restore procedure correctly.

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 {
 READ NAMES[X] = (OL1[X]), Object_Name
 }
5. IF (Protocol_Revision is present and Protocol_Revision \geq 10) THEN
 READ BPT = Backup_Preparation_Time
 READ RPT = Restore_Preparation_Time
 READ RCT = Restore_Completion_Time
 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 \geq 10) THEN
 WAIT BPT
 READ BRSTATE = Backup_And_Restore_State
 READ CF = Configuration_Files
 WHILE (BRSTATE = PREPARING_FOR_BACKUP) DO {
 WAIT 1 second

```

        READ BRSTATE = Backup_And_Restore_State
        IF CF is an empty list THEN
            READ CF = Configuration_Files
        IF CF is a non-empty list THEN
            READ X = (the file referenced by
Configuration_Files[1]).Name
        }
        CHECK (BRSTATE = PERFORMING_A_BACKUP)
    9. READ CF = Configuration_Files
    10. CHECK (CF is a non-empty array of BACnetObjectIdentifiers referring to File
objects)
    11. REPEAT X = (each entry in CF) DO {
        READ Y = X, File_Access_Method
        IF (Y = RECORD_ACCESS)
            WHILE (the last read resulted in an Ack with 'End Of File' ==
FALSE) DO {
                TRANSMIT AtomicReadFile-Request,
                    'Object Identifier' = X,
                    'File Start Record' = (the next
unread record),
                    'Requested Record Count' = 1
                RECEIVE AtomicReadFile-ACK,
                    'End Of File' = TRUE | FALSE,
                    'File Start Record' = Z,
                    'Requested Record Count' = 1
                    'Returned Data' = (File contents)
                | Error-PDU -- only acceptable for the first record
and only when there are no records in the file
                    'Error Class' = SERVICES,
                    'Error Code' =
INVALID_FILE_START_POSITION
                }
            ELSE
                WHILE (the last read did not indicate 'End Of File') DO {
                    TRANSMIT AtomicReadFile-Request,
                        'Object Identifier' = X,
                        'File Start Position' = (the next unread
octet),
                        'Requested Octet Count' = MROC
                    RECEIVE AtomicReadFile-ACK,
                        'End Of File' = TRUE | FALSE,
                        'File Start Position' = (the next unread octet)
                        'File Data' = (File contents of length MROC
if 'End Of File' is FALSE
or of length MROC or
less if 'End Of File' is TRUE)

```

| Error-PDU -- only acceptable for the first record
and only when there are no records in the file

'Error Class' = SERVICES,

'Error Code' =

```

INVALID_FILE_START_POSITION
    }
}
12. TRANSMIT ReinitializeDevice-Request,
    'Reinitialize State Of Device' = ENDBACKUP,
    'Password' = (any valid password)
13. RECEIVE BACnet-Simple-ACK-PDU
14. VERIFY System_Status != BACKUP_IN_PROGRESS
15. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) 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
    READ BRSTATE = Backup_And_Restore_State
    WHILE (BRSTATE = PREPARING_FOR_RESTORE) DO {
        WAIT 1 second
        READ BRSTATE = Backup_And_Restore_State
    }
    CHECK (BRSTATE = PERFORMING_A_RESTORE)
20. READ OL2 = Object_List
21. REPEAT X = (entry in CF) DO {
    IF (X is not in OL2)
        TRANSMIT CreateObject-Request
            'Object Identifier' = X
        RECEIVE CreateObject-ACK
            'Object Identifier' = X
    READ FS = X, File_Size
    IF (File_Size is not equal to the size of the backed up file)
        WRITE X, File_Size = 0
    IF (Y = RECORD_ACCESS)
        TRANSMIT AtomicWriteFile-Request
            'File Identifier' = X
            'File Start Record' = 0

```

obtained in step 11)

```

        'Record Data' = (file content for first record
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' = -1
}
ELSE
    REPEAT Z = (0 through the file size, in increments of MWDL)
DO {
    TRANSMIT AtomicWriteFile-Request
        'File Identifier' = X
        'File Start Position' = Z
        'Record Data' = (file contents obtained from
the backup, the number of octets
being the
lesser of (file size - Z) and MWDL)
    RECEIVE AtomicWriteFile-ACK
        'File Start Position' = Z
}
}
22. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
    VERIFY Backup_And_Restore_State = RESTORE_IN_PROGRESS
23. TRANSMIT ReinitializeDevice-Request,
    'Reinitialize State Of Device' = ENDRESTORE,
    'Password' = (any valid password)
24. RECEIVE BACnet-Simple-ACK-PDU
25. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
    WAIT RCT
    VERIFY Backup_And_Restore_State = IDLE
26. READ DR3 = Database_Revision
27. CHECK (DR3 <> DR1)
28. IF (Database_Revision was changed in step 16) THEN
    CHECK (DR3 <> DR2)
29. VERIFY Last_Restore_Time > LRT1
30. READ OL3 = Object_List
31. CHECK (that OL1 and OL3 contain the same set of objects)
32. REPEAT X = (1 through length of OL1) DO {
    VERIFY (OL1[X]), Object_Name = NAMES[X]
}

```

13.8.1.2 Attempting a Backup Procedure While Already Performing a Backup Procedure

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

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

Test Steps:

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

13.8.1.3 Attempting a Backup Procedure While Already Performing a Restore Procedure

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

Test Steps:

1. IF (Protocol_Revision is present and Protocol_Revision \geq 10) THEN

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

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

13.8.1.4 Attempting a Restore Procedure While Already Performing a Backup Procedure

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

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

Test Steps:

1. IF (Protocol_Revision is present and Protocol_Revision \geq 10) THEN
 READ BPT = Backup_Preparation_Time
2. TRANSMIT ReinitializeDevice-Request,
 'Reinitialized State of Device' = STARTBACKUP,
 'Property Identifier' = (any valid password)
3. RECEIVE Simple-ACK-PDU
4. IF (Protocol_Revision is present and Protocol_Revision \geq 10) THEN
 WAIT BPT
5. TRANSMIT
 SNET = (N, any remote network number),

- SADR = (M, any MAC address valid for the specified network),
 ReinitializeDevice-Request,
 'Reinitialized State of Device' = STARTRESTORE,
 'Property Identifier' = (any valid password),
6. RECEIVE
 - DNET = N,
 - DADR = M,
 - BACnet-Error PDU,
 - Error Class = DEVICE,
 - Error Code = CONFIGURATION_IN_PROGRESS
 7. TRANSMIT ReinitializeDevice-Request,
 'Reinitialized State of Device' = ENDBACKUP,
 'Property Identifier' = (any valid password)
 8. RECEIVE Simple-ACK-PDU

13.8.1.5 Attempting a Restore Procedure While Already Performing a Restore Procedure

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

Test Steps:

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

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

13.8.1.6 Ending Backup and Restore Procedures via Timeout

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

Test Steps:

1. WRITE Backup_Failure_Timeout = (A value T1 greater than Backup_Preparation_Timeout)
2. VERIFY Backup_Failure_Timeout = T1
3. IF (Protocol_Revision is present and Protocol_Revision \geq 10) THEN
 READ BPT = Backup_Preparation_Time
4. TRANSMIT ReinitializeDevice-Request,
 'Reinitialized State of Device' = STARTBACKUP,
 'Property Identifier' = (any valid password)
5. RECEIVE Simple-ACK-PDU
6. IF (Protocol_Revision is present and Protocol_Revision \geq 10) THEN
 WAIT BPT
 READ BRSTATE = Backup_And_Restore_State
 WHILE (BRSTATE = PREPARING_FOR_BACKUP) DO {
 WAIT 1 second
 READ BRSTATE = Backup_And_Restore_State
 }
 CHECK (BRSTATE = PERFORMING_A_BACKUP)
7. WAIT (T1 + 10 seconds)
8. IF (Protocol_Revision is present and Protocol_Revision \geq 10) THEN
 VERIFY Backup_And_Restore_State = IDLE
9. VERIFY System_Status != BACKUP_IN_PROGRESS
10. IF (Protocol_Revision is present and Protocol_Revision \geq 10) THEN
 READ RPT = Restore_Preparation_Time
 READ RCT = Restore_Completion_Time
11. TRANSMIT ReinitializeDevice-Request,
 'Reinitialized State of Device' = STARTRESTORE,
 'Password' = (any valid password)
12. RECEIVE BACnet-Simple ACK-PDU
13. IF (Protocol_Revision is present and Protocol_Revision \geq 10) THEN
 WAIT RPT
 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)
14. WAIT (40 seconds)
15. IF (Protocol_Revision is present and Protocol_Revision ≥ 10) THEN
 WAIT RCT
 VERIFY Backup_And_Restore_State = IDLE
16. VERIFY System_Status != DOWNLOAD_IN_PROGRESS

```

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

### 13.8.1.7 Ending Backup and Restore Procedures via Abort

Purpose: This test case verifies that the IUT will leave the BACKUP\_IN\_PROGRESS and DOWNLOAD\_IN\_PROGRESS states upon a command to abort.

Test Steps:

1. IF (Protocol\_Revision is present and Protocol\_Revision ≥ 10) THEN  
    READ BPT = Backup\_Preparation\_Time
2. TRANSMIT ReinitializeDevice-Request,  
    'Reinitialized State of Device' = STARTBACKUP,  
    'Password' = (any valid password)
3. RECEIVE BACnet-Simple ACK-PDU
4. IF (Protocol\_Revision is present and Protocol\_Revision ≥ 10) THEN  
    WAIT BPT
5. TRANSMIT ReinitializeDevice-Request,  
    'Reinitialized State of Device' = ENDBACKUP,  
    'Password' = (any valid password)
6. RECEIVE BACnet-Simple ACK-PDU
7. IF (Protocol\_Revision is present and Protocol\_Revision ≥ 10) THEN  
    VERIFY Backup\_And\_Restore\_State = IDLE
8. VERIFY System\_Status != BACKUP\_IN\_PROGRESS
9. IF (Protocol\_Revision is present and Protocol\_Revision ≥ 10) THEN  
    READ RPT = Restore\_Preparation\_Time  
    READ RCT = Restore\_Completion\_Time
10. TRANSMIT ReinitializeDevice-Request,  
    'Reinitialized State of Device' = STARTRESTORE,  
    'Password' = (any valid password)
11. RECEIVE BACnet-Simple ACK-PDU
12. IF (Protocol\_Revision is present and Protocol\_Revision ≥ 10) THEN  
    WAIT RPT
13. TRANSMIT ReinitializeDevice-Request,  
    'Reinitialized State of Device' = ABORTRESTORE,

- 'Password' = (any valid password)
14. RECEIVE BACnet-Simple ACK-PDU
  15. IF (Protocol\_Revision is present and Protocol\_Revision  $\geq$  10) THEN  
    WAIT RCT  
    VERIFY Backup\_And\_Restore\_State = IDLE
  16. VERIFY System\_Status != DOWNLOAD\_IN\_PROGRESS

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

#### **13.8.1.8 Attempting a Backup Procedure with an Invalid Password**

Purpose: To verify the correct execution of the Backup procedure when an invalid password is provided. If the IUT cannot be made to deny a ReinitializeDevice <STARTBACKUP> service request that does not contain a valid password, then this test shall be omitted.

Test Steps:

1. TRANSMIT ReinitializeDevice-Request,  
    'Reinitialized State of Device' = STARTBACKUP,  
    'Password' = (any invalid password)
2. RECEIVE BACnet-Error-PDU,  
    Error Class = SECURITY,  
    Error Code = PASSWORD\_FAILURE

#### **13.8.1.9 Attempting a Restore Procedure with an Invalid Password**

Purpose: To verify the correct execution of the Restore procedure when an invalid password is provided. If the IUT cannot be made to deny a ReinitializeDevice <STARTRESTORE > service request that does not contain a valid password, then this test shall be omitted.

Test Steps:

1. TRANSMIT ReinitializeDevice-Request,  
    'Reinitialized State of Device' = STARTRESTORE,  
    'Password' = (any invalid password)
2. RECEIVE BACnet-Error-PDU,  
    Error Class = SECURITY,  
    Error Code = PASSWORD\_FAILURE

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

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

Test Steps:

1. IF (Protocol\_Revision is present and Protocol\_Revision  $\geq$  10) THEN  
    READ BPT = Backup\_Preparation\_Time
2. TRANSMIT ReinitializeDevice-Request,  
    'Reinitialized State of Device' = STARTBACKUP,  
    'Password' = (any non-zero length password)
3. RECEIVE BACnet-Simple ACK-PDU
4. IF (Protocol\_Revision is present and Protocol\_Revision  $\geq$  10) THEN  
    WAIT BPT
5. TRANSMIT ReinitializeDevice-Request,  
    'Reinitialized State of Device' = ENDBACKUP,  
    'Password' = (any non-zero length password)
6. RECEIVE BACnet-Simple ACK-PDU
7. IF (Protocol\_Revision is present and Protocol\_Revision  $\geq$  10) THEN  
    VERIFY Backup\_And\_Restore\_State = IDLE
8. VERIFY System\_Status  $\neq$  BACKUP\_IN\_PROGRESS

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

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

Test Steps:

1. IF (Protocol\_Revision is present and Protocol\_Revision  $\geq$  10) THEN  
    READ RPT = Restore\_Preparation\_Time  
    READ RCT = Restore\_Completion\_Time
2. TRANSMIT ReinitializeDevice-Request,  
    'Reinitialized State of Device' = STARTRESTORE,  
    'Password' = (any non-zero length password)
3. RECEIVE BACnet-Simple ACK-PDU
4. IF (Protocol\_Revision is present and Protocol\_Revision  $\geq$  10) THEN  
    WAIT RPT
5. TRANSMIT ReinitializeDevice-Request,  
    'Reinitialized State of Device' = ENDRESTORE,  
    'Password' = (any non-zero length password)
6. RECEIVE BACnet-Simple ACK-PDU
7. IF (Protocol\_Revision is present and Protocol\_Revision  $\geq$  10) THEN  
    WAIT RCT

8. VERIFY System\_Status != DOWNLOAD\_IN\_PROGRESS

#### 13.8.1.12 System\_Status during a Backup Procedure

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

Test Steps:

1. IF (Protocol\_Revision is present and Protocol\_Revision  $\geq$  10) THEN  
    READ BPT = Backup\_Preparation\_Time
2. TRANSMIT ReinitializeDevice-Request,  
    'Reinitialized State of Device' = STARTBACKUP,  
    'Password' = (any valid password)
3. RECEIVE BACnet-Simple ACK-PDU
4. IF (Protocol\_Revision is present and Protocol\_Revision  $\geq$  10) THEN  
    WAIT BPT
5. VERIFY System\_Status = BACKUP\_IN\_PROGRESS
6. TRANSMIT ReinitializeDevice-Request,  
    'Reinitialized State of Device' = ENDBACKUP,  
    'Password' = (any valid password)
7. RECEIVE BACnet-Simple ACK-PDU
8. WAIT a vendor specified period of time for the device to complete the backup operation
9. VERIFY System\_Status != BACKUP\_IN\_PROGRESS

#### 13.8.1.13 System\_Status during a Restore Procedure

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

Test Steps:

1. IF (Protocol\_Revision is present and Protocol\_Revision  $\geq$  10) THEN  
    READ RPT = Restore\_Preparation\_Time  
    READ RCT = Restore\_Completion\_Time
2. TRANSMIT ReinitializeDevice-Request,  
    'Reinitialized State of Device' = STARTRESTORE,  
    'Password' = (any valid password)
3. RECEIVE BACnet-Simple ACK-PDU
4. IF (Protocol\_Revision is present and Protocol\_Revision  $\geq$  10) THEN  
    WAIT RPT
5. VERIFY System\_Status = DOWNLOAD\_IN\_PROGRESS

6. TRANSMIT ReinitializeDevice-Request,  
     'Reinitialized State of Device' = ABORTRESTORE,  
     'Password' = (any valid password)
7. RECEIVE BACnet-Simple ACK-PDU
8. IF (Protocol\_Revision is present and Protocol\_Revision  $\geq$  10) THEN  
     WAIT RCT
9. VERIFY System\_Status != DOWNLOAD\_IN\_PROGRESS

### **13.8.2 Backup and Restore Initiation Tests**

The tests in this section verify that a device which can back up and restore other devices has implemented the initiation of the Backup and Restore procedure correctly.

The tests in this section rely on the TD being able to emulate a device that can be backed up and restored. The ability to configure the characteristics of the TD with respect to the Backup and Restore operations is important to ensure adequate testing of the IUT.

#### **13.8.2.1 Initiate a Full Backup and Restore**

Purpose: To verify that the IUT can perform a Backup and Restore on a BACnet server device.

Test Concept: The IUT is first made to initiate a Backup and then a Restore of the TD device. This test verifies that the IUT performs the Backup procedure correctly by comparing the resulting restored file with the original. The TD is made to respond appropriately such that the Backup and Restore procedures are completed normally. The final check can be accomplished using a file compare of the original files to the files restored or by comparing the network traffic during the backup to the network traffic during the restore. The number of files, the order of the files, and the file content should be the same. The test is to be executed multiple times with the TD configured with different sets of backup and restore characteristics.

Configuration Requirements: The IUT is configured to already contain a device binding for the TD device. The TD is configured with some of the following characteristics:

Backup Characteristics:

1. The TD is configured to contain an APDU size that is smaller than the APDU size of the IUT. If the TD and the IUT support segmentation, the TD is configured to support a smaller window size than the IUT.
2. The TD is configured to contain a configuration file of size zero.
3. The TD is configured to contain some configuration files that are STREAM\_ACCESS and some that are RECORD\_ACCESS.

4. The TD is configured to only allow access to File and Device objects during the Backup and Restore procedures. All other attempts shall result in an error from the TD.
5. The TD is configured to require the same password for all of the reinitialize device requests.
6. The TD is configured to contain file names that would not be accepted by the OS which the IUT is running on.
7. The TD is configured with a Protocol\_Revision < 10.
8. The TD is configured with a Protocol\_Revision  $\geq 10$ . This is only used if the IUT claims Protocol\_Revision  $\geq 10$ .

#### Restore Characteristics:

1. The TD is configured to support CreateObject service, and some of the configuration files exist while others do not.
2. The TD is configured such that some of the configuration file File objects exist, but the file size is different from that of the file to be restored.
3. The TD is configured to not support the CreateObject service.
4. The TD is configured to contain some configuration files that are STREAM\_ACCESS and some that are RECORD\_ACCESS.
5. The TD is configured to only allow access to File and Device objects during the Backup and Restore procedures. All other attempts shall result in an error from the TD.
6. The TD is configured to require the same password for all of the reinitialize device requests.
7. The TD is configured with a Protocol\_Revision < 10.
8. The TD is configured with a Protocol\_Revision  $\geq 10$ . This is only used if the IUT claims Protocol\_Revision  $\geq 10$ .

#### Test Steps:

1. MAKE (IUT initiate a backup on the TD device)
2. WAIT (for backup to complete)
3. MAKE (changes required in TD to meet restore characteristics for this test)
4. MAKE (IUT initiate a restore on the TD device)
5. WAIT (for restore to complete)
6. CHECK (that the file content restored is the same as the file content that was backed up)

#### Notes to Tester: Other items to ensure were correct during execution of the test:

1. Verify the order the IUT read the configuration files was the same as the order returned by the Configuration\_Files property.
2. Verify that any file with a File\_Size of zero was restored.
3. Verify that each file read is in byte order if STREAM\_ACCESS and in record order if RECORD\_ACCESS.

### 13.8.2.2 Can Abort Backup if Error Received from TD

**Purpose:** To verify that the IUT can abort the Backup procedure when an error is received from the TD during a backup attempt.

**Test Concept:** The IUT is made to initiate a backup of the TD device. Before the backup is completed, the TD shall return an error. This error will be a BACnetAbortPDU, BACnetErrorPDU, or BACnetRejectPDU response to one of the confirmed requests initiated by the IUT during the backup process. When the IUT receives this error, it should abort the backup process.

**Configuration Requirements:** The IUT is configured to already contain a device binding for the TD device. For this test to be performed, the TD shall be able to return an error for one of the confirmed requests initiated by the IUT during a backup.

**Test Steps:**

1. MAKE (IUT start backup on TD)
2. WAIT (until a tester chosen point in the backup)
3. WHILE (a ReinitializeDevice-Request is not received) DO {
 

IF (confirmed request received) THEN  
   TRANSMIT  
     BACnet-AbortPDU,  
     AbortReason = (any value)  
     | (BACnet-ErrorPDU,  
     Error Class = OBJECT | PROPERTY,  
     Error Code = (any of the error codes for an  
     OBJECT or PROPERTY class)  
     | (BACnet-Reject PDU,  
     Reject Reason = (any value) )
- }
4. RECEIVE ReinitializeDevice-Request,  
    'Reinitialize State Of Device' = ENDBACKUP,  
    'Password' = (any valid password)
5. TRANSMIT BACnet-Simple-ACK-PDU

**Notes to Tester:** An IUT that never stops the Backup procedure after several attempts by the TD to return an error shall fail this test.

### 13.8.2.3 Can Abort Restore if Error Received from TD

**Purpose:** To verify that the IUT can abort the restore procedure when an error is received from the TD during a restore attempt.



Test Concept: The IUT is made to initiate a restore of the TD device. Before the restore is completed, the TD returns an error. This error will be a BACnet Abort PDU or BACnet Reject PDU response to one of the confirmed requests initiated by the IUT during the backup process. When the IUT receives this error, it should abort the restore process.

Configuration Requirements: The IUT is configured to already contain a device binding for the TD device. For this test to be performed, the TD shall be able to return an error for one of the confirmed requests initiated by the IUT during a restore.

Test Steps:

1. MAKE (IUT start restore on TD)
2. WAIT (until a tester selected time during the restore procedure)
3. WHILE (a ReinitializeDevice-Request is not received) DO {
 

IF (confirmed request received) THEN
 TRANSMIT
 

BACnet-AbortPDU,  
 AbortReason = (any value)  
 | (BACnet-ErrorPDU,  
 Error Class = OBJECT | PROPERTY,  
 Error Code = (any of the error codes for an  
 OBJECT or PROPERTY class) )  
 | (BACnet-Reject PDU,  
 Reject Reason = (any value) )
4. RECEIVE ReinitializeDevice-Request,  
     'Reinitialize State Of Device' = ABORTRESTORE,  
     'Password' = (any valid password)
5. TRANSMIT BACnet-Simple-ACK-PDU

Notes to Tester: An IUT that never stops the restore procedure after several attempts by the TD to return an error shall fail this test.

### **13.8.2.5 Initiate an Abort Backup**

Purpose: To verify that the IUT can abort the Backup procedure.

Test Concept: The IUT is made to initiate a backup of the TD device. Before the backup is completed, the IUT requests an abort of the backup.

Configuration Requirements: The IUT is configured to already contain a device binding for the TD device. A TD that takes a long time to backup is required.

Test Steps:

1. MAKE (IUT start backup on TD)

2. BEFORE (backup is complete)  
    MAKE (IUT end backup on TD)
3. RECEIVE ReinitializeDevice-Request,  
    'Reinitialize State Of Device' = ENDBACKUP,  
    'Password' = (any valid password)
4. TRANSMIT BACnet-Simple-ACK-PDU

#### **13.8.2.6 Initiate an Abort Restore**

Purpose: To verify that the IUT can abort the restore procedure.

Test Concept: The IUT is made to initiate a restore of the TD device. Before the restore is completed, the IUT requests an abort of the restore procedure.

Configuration Requirements: The IUT is configured to already contain a device binding for the TD device.

Test Steps:

1. MAKE (IUT start restore on TD)
2. BEFORE ( restore is complete )  
    MAKE (IUT abort restore on TD)
3. RECEIVE ReinitializeDevice-Request,  
    'Reinitialize State Of Device' = ABORTRESTORE,  
    'Password' = (any valid password)
4. TRANSMIT BACnet-Simple-ACK-PDU