

Clarification Request

References: ASHRAE 135.1-2023, ASHRAE 135-2024, Test Package 26.0.

Date of BTL-WG Response: September 4, 2025

Background:

12.13.6 File_Size

This property, of type Unsigned, indicates the size of the file data in octets. If the size of the file can be changed by writing to the file, and File_Access_Method is STREAM_ACCESS, then this property shall be writable.

...

Devices may restrict the allowed values for writes to the File_Size. Specifically, devices may allow deletion of the file contents by writing a value of zero, but not necessarily allow arbitrary truncation or expansion.

...

3.61.5 Supports a Stream-Based File Object for a Purpose Other Than Backup and Restore or BACnet/SC Certificate Exchange

The IUT supports a data stream-based File that is not accessed during Backup and Restore or BACnet/SC Certificate Exchange.

135.1-2023 - 9.12.1.2.1 - Reading an Entire Stream-Based File		
	Test Conditionality	Must be executed.
	Test Directives	Apply to a file not used for Backup & Restore or BACnet/SC Certificate Exchange.
	Testing Hints	
...		

9.12.1.2.1 Reading an Entire Stream Based File

Purpose: To verify that the IUT correctly responds to a request to read an entire file.

Test Concept: The test consists of reading the contents of the file using a sequence of AtomicReadFile requests and verifying that the appropriate known file data is returned.

Configuration Requirements: The AtomicReadFile service execution tests require that the TD has knowledge of the exact contents of a known file F1. The test procedures assume that the IUT is already configured with the known file data provided by the manufacturer. In the test procedures "X" will designate the File object identifier and Z the 'File Start Position' initialized at "0". When performing the AtomicReadFile services, a Maximum Requested Octet Count (MROC) 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.

Test Steps:

1. VERIFY File_Access_Method = STREAM_ACCESS
2. WHILE (the last read resulted in an Ack with 'End Of File' = FALSE) DO {
TRANSMIT AtomicReadFile-Request,
'Object Identifier' = X,

- ```

 'File Start Position' = Z (the next unread octet),
 'Requested Octet Count' = MROC
 RECEIVE AtomicReadFile-ACK,
 'End Of File' = TRUE | FALSE,
 'File Start Position' = Z
 'File Data' = (the known contents of the test file of length MROC if 'End Of File' is
 FALSE or of length MROC or less if 'End Of File' is TRUE)
 }
3. CHECK(that the returned file data is F1)

```

### 3.61.6 Contains a Writable Stream-Based File for a Purpose Other Than Backup and Restore or BACnet/SC Certificate Exchange

The IUT supports a data stream-based File that is not accessed during Backup and Restore or BACnet/SC Certificate Exchange.

| 135.1-2023 - 9.13.1.2.1 - Writing an Entire Stream-Based File |                     |                                                                                  |
|---------------------------------------------------------------|---------------------|----------------------------------------------------------------------------------|
|                                                               | Test Conditionality | Must be executed.                                                                |
|                                                               | Test Directives     | Apply to a file not used for Backup & Restore or BACnet/SC Certificate Exchange. |
|                                                               | Testing Hints       |                                                                                  |
| ...                                                           |                     |                                                                                  |

#### 9.13.1.2.1 Writing an Entire Stream Based File

Purpose: To verify that the IUT correctly responds to a request to write an entire file.

Test Concept: The tests consist of modifying the contents of the files using the AtomicWriteFile service and verifying that the appropriate changes to the file data took place

Configuration Requirements: The manufacturer shall provide appropriate test data to write to these files or sufficient information to permit the tester to construct the test data. The file objects shall be configured with initial data that differs from the test data. In the test procedures, "X" will designate the File object identifier and Z the File Start Position' initialized at "1" at the beginning. When performing the AtomicWriteFile services, a Maximum Write Data Length (MWDL) shall be calculated before starting the test. These values shall be used during the test. 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. VERIFY Read\_Only = FALSE
2. WRITE Archive = TRUE
3. VERIFY File Access Method = STREAM ACCESS
4. IF (File\_Size is not equal to the size of the test file) THEN  
WRITE File\_Size = 0
5. REPEAT Z = (0 through the file size, in increments of MWDL) DO {  
TRANSMIT AtomicWriteFile-Request  
'File Identifier' = X  
'File Start Position' = Z  
'File Data' = (file contents, the number of octets being the lesser of (file size - Z)  
and MWDL)  
RECEIVE AtomicWriteFile-ACK  
'File Start Position' = Z

- ```
}  
6. VERIFY File_Size = (file size of the test data)  
7. VERIFY Modification_Date = (the current date and time)  
8. VERIFY ARCHIVE = FALSE
```

Problem:

A vendor uses AtomicWriteFile to write a file to a device. The file is not a Backup/Restore file or a BACnet/SC certificate. The file is absorbed into the device as it is written and cannot be read back from the device. Thus, the File_Size is read-only and always 0.

Clause 12.13.6 requires the File_Size to be writable if the size of the file is changed by writing to the file. This allows File_Size to have a fixed value for a fixed sized file. This clause also allows the property to support limited values including only the value 0.

Since File_Size is always 0, test 9.12.1.2.1 cannot be executed.

Step #4 in test 9.13.1.2.1 expects File-Size to be written to zero.

Step #6 in test 9.13.1.2.1 expects File_Size to equal the size of the data sent.

Question #1:

Can File_Size be read-only if it can only equal 0?

Response #1:

Yes

Question #2:

Can Test 9.12.1.2.1 be skipped if File_Size can only equal 0?

Response #2:

Yes

Question #3:

Can Step #4 in Test 9.13.1.2.1 be skipped if File_Size equals 0?

Response #3:

Yes

Question #4:

Can Step #6 in Test 9.13.1.2.1 allow for File_Size to equal 0?

Response #4:

Yes