BACnet® TESTING LABORATORIES
ADDENDA

Addendum s to
BTL Test Package 15

Revision 3
Revised 8/7/2019

Approved by the BTL Working Group on June 20, 2019;
Approved by the BTL Working Group Voting Members on August 6, 2019;
Published on August 7, 2019.

[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 15.2 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-15.2s-1: File Object Testing [BTLWG-427]

Overview:
To create new tests for File (Protocol Rev 2 object type)

Changes:
[In BTL Checklist, modify File object type in Section 3, Objects]

Support	Listing	Option
File Object		
	R	Base Requirements
	C[1]	Supports DM-BR-B
	~~C[1]~~	~~Supports a File object for a purpose other than Backup and Restore~~
	~~O~~	~~Contains a writable File for a purpose other than Backup and Restore~~
	C[1]	*Supports a record-based File object for a purpose other than Backup and Restore*
	C[1]	*Supports a stream-based File object for a purpose other than Backup and Restore*
	O	*Contains a writable stream-based File for a purpose other than Backup and Restore*
[1] At least one of these options is required if the IUT supports the File object type.		

[In BTL Test Plan 15.2, modify File object tests in section 3.61]

Objects

3.61 File

3.61.1 Base Requirements

For File object, there are no base requirements.

3.61.2 Supports DM-BR-B

The IUT supports a data File that is readable and writable during Backup and Restore using AtomicReadFile and AtomicWriteFile requests.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for Device Management - Backup and Restore - B in the Checklist.
	Testing Hints	

~~3.61.3 Supports a File object for a purpose other than Backup and~~ Restore

~~For a device which contains a File object for a purpose other than Backup and Restore, there are no testing requirements.~~

~~3.61.4Contains a writable File for a purpose other than Backup and Restore~~

~~For a device which contains a writable File object for a purpose other than Backup and Restore, there are no testing requirements.~~

3.61.3 Supports a Record-Based File Object for a Purpose Other Than Backup and

Restore

For a device which contains a record-based File object for a purpose other than Backup and Restore, there are no testing requirements.

3.61.4 Supports a Stream-Based File Object for a Purpose Other Than Backup and Restore

The IUT supports a data stream-based File that is not accessed during Backup and Restore.

BTL - 9.12.1.2.1 - Reading an Entire Stream-based File		
	Test Conditionality	*Must be executed.*
	Test Directives	
	Testing Hints	
135.1-2013 - 9.12.2.2.2 - Attempting to Read Data from a Nonexistent File		
	Test Conditionality	*Must be executed.*
	Test Directives	
	Testing Hints	
135.1-2013 - 9.12.2.2.3 - Attempting to Read Data Using the Wrong File Access Type		
	Test Conditionality	*Must be executed.*
	Test Directives	
	Testing Hints	

3.61.5 Contains a Writable Stream-Based File for a Purpose Other Than Backup and Restore

The IUT supports a data stream-based File that is not accessed during Backup and Restore.

BTL - 9.13.1.2.1 - Writing an Entire Stream-based File		
	Test Conditionality	*Must be executed.*
	Test Directives	
BTL - 9.13.1.2.3 - Appending Data to the End of a File		
	Test Conditionality	*If the file size cannot be changed or if the IUT does not support files that cannot be modified except by replacing the entire file, then this test shall be skipped.*
	Test Directives	
	Testing Hints	
135.1-2013 - 9.13.1.2.4 - Truncating a File		
	Test Conditionality	*If the file size cannot be changed, then this test shall be skipped.*
	Test Directives	
	Testing Hints	
135.1-2013 - 9.13.1.2.5 - Deleting a File		
	Test Conditionality	*If the file size cannot be changed, then this test shall be skipped.*
	Test Directives	
	Testing Hints	
135.1-2013 - 9.13.2.2.1 - Writing to a Stream Access File using Record Access		
	Test Conditionality	*Must be executed.*
	Test Directives	
	Testing Hints	
135.1-2013 - 9.13.2.2.2 - Writing to a File with an Invalid Starting Position		
	Test Conditionality	*Must be executed.*
	Test Directives	
	Test Directives	
135.1-2013 - 9.13.2.2.4 - Writing to a Nonexistent File		
	Test Conditionality	*Must be executed.*
	Test Directives	
	Testing Hints	

[In BTL Specified Tests, add tests, as shown. The entirely new tests are shown without *italics* or ~~strike-through~~.]

[Replace test 135.1 - 9.12.1.2.1 with a completely new test and add to BTL Specified Tests]

9.12.1.2.1 Reading an Entire *Stream-Based* File

Reason for Change: Change to allow testing of files larger than that which can be returned in a single request.

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. TRANSMIT AtomicReadFile-Request,~~
~~'File Identifier' = S,~~
~~'File Start Position' = 0,~~
~~'Requested Octet Count' = (the number of octets in the test file)~~
~~2. RECEIVE AtomicReadFile-ACK,~~
~~'End of File' = TRUE,~~
~~'File Start Position' = 0,~~
~~'File Data' = (the known contents of the test file)~~

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)

[Replace test 135.1 - 9.13.1.2.1 with a completely new test and add to BTL Specified Tests]

9.13.1.2.1 Writing an Entire *Stream-Based* File

Reason for Change: Allow the entire file content to be written in all cases.

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 test data shall contain at least as many octets as the initial data for the file.~~ *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. TRANSMIT AtomicReadFile-Request,~~
~~'File Identifier' = S,~~
~~'File Start Position = 0,~~
~~'Requested Octet Count' = (any number ≥ the number of octets in the test data)~~
~~2. RECEIVE AtomicReadFile-ACK,~~
~~'End of File' = TRUE,~~
~~'File Start Position' = 0,~~
~~'File Record Data' = (the initial data)~~
~~3. TRANSMIT AtomicWriteFile-Request,~~
~~'File Identifier' = S,~~
~~'File Start Position = 0,~~
~~'File Data' = (the test data)~~
~~4. RECEIVE AtomicWriteFile-ACK,~~
~~'File Start Position' = 0~~
~~5. TRANSMIT AtomicReadFile-Request,~~
~~'File Identifier' = S,~~
~~'File Start Position' = 0,~~
~~'Requested Octet Count' = (any number > the number of octets in the test data)~~
~~6. RECEIVE AtomicReadFile-ACK,~~
~~'End of File' = TRUE,~~
~~'File Start Position' = 0,~~
~~'File Data' = (the test data)~~
~~7. VERIFY (R), Modification_Date = (the current date and time)~~
~~8. VERIFY (R), ARCHIVE = FALSE~~
~~9. VERIFY (R), Number_Of_Records = (the number of records in the test data)~~

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

[Modify 135.1 - 9.13.1.2.3 and add to BTL Specified Tests]

9.13.1.2.3 Appending Data to the End of a File

Reason for Change: Added configuration requirements, fixed purpose, and removed reliance on AtomicReadFile within the test steps.

Purpose: To verify that the IUT correctly responds to a request to write to the end of a file. ~~If the IUT does not support files that can not be modified except by replacing the entire file, and this restriction is clearly stated in the PICS, then this test may be ignored.~~

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. 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. TRANSMIT ReadProperty-Request,
   'Object Identifier' = S,
   'Property Identifier' = File_Size
2. RECEIVE ReadProperty-ACK,
   'Object Identifier' = S,
   'Property Identifier' = File_Size,
   'Property Value' = (the current size in octets, designated "InitialNumOctets" below)

~~3. TRANSMIT AtomicReadFile-Request,~~
~~'File Identifier' = S,~~
~~'File Start Position' = 0,~~
~~'Requested Octet Count' = (any number > InitialNumOctets)~~
~~4. RECEIVE AtomicReadFile-ACK,~~
~~'End of File' = TRUE,~~
~~'File Start Position' = 0,~~
~~'File Data' = (the initial data)~~

3. TRANSMIT AtomicWriteFile-Request,
   'File Identifier' = S,
   'File Start Position' = -1,
   'File Data' = (the test data*, the number of octets being the lesser of (file size - Z) and MWDL*)
4. RECEIVE AtomicWriteFile-ACK,
   'File Start Position' = InitialNumOctets,

~~7. TRANSMIT AtomicReadFile-Request,~~
~~'File Identifier' = S,~~
~~'File Start Position' = InitialNumOctets,~~
~~'Requested Octet Count' = (the number of octets in the test data)~~
~~8. RECEIVE AtomicReadFile-ACK,~~
~~'End of File' = TRUE,~~
~~'File Start Position' = InitialNumOctets,~~
~~'File Data' = (the test data)~~

5. VERIFY (R), Modification_Date = (the current date and time)
6. VERIFY (R), ARCHIVE = FALSE
7. VERIFY (R), File_Size = (the number of octets in the test data + InitialNumOctets)

BTL-15.2s-2: Add Testing for Extended MSTP Frames [BTLWG-560]

Overview:
Addendum 2012an-1 made provisions for extended MSTP frames with NPDU sizes up to 1497 octets. This work item adds test coverage for MS/TP nodes that support extended MS/TP frames and for routers that can be directly connected to MS/TP networks.

Changes:
[In BTL Checklist, add entries in Section 9 Data Link Layer]

Data Link Layer - MS/TP - Master Node		
	R	Base Requirements
	C[1]	Supports writable Max_Master property
	C[1]	Supports read only Max_Master property
	C[2]	Contains configurable Max_Info_Frames property
	C[2]	Contains non-configurable Max_Info_Frames property
	O	Is a BACnet router
	O[3]	*Supports extended MS/TP frames (over 501 octets)*
[1] Exactly one of these options is required in order to claim conformance to this BIBB. [2] Exactly one of these options is required in order to claim conformance to this BIBB. *[3]Protocol_Revision 16 or higher must be claimed*		

[In BTL Test Plan, add entries for Supports Extended Frames (over 501 octets) in section 9]

9.1.7 Supports Extended MS/TP Frames (over 501 octets)

The IUT can transmit and receive messages with an NPDU > 501 octets

BTL - 12.1.3.X1 - Frame Type Based on Transmitted NPDU Size		
	Test Conditionality	Must be executed
	Test Directives	Execute the test such that the transmitted NPDU sizes are near the 501 octet boundary.
	Testing Hints	
BTL - 12.1.3.X2 - Executing COBS Encoded Frames		
	Test Conditionality	Must be executed
	Test Directives	
	Testing Hints	

[In BTL Specified Tests, add two new tests]

12.1.3.X1 Frame Type Based on Transmitted NPDU size

Purpose: To verify that the IUT selects the correct frame type based on the transmitted NPDU size.

Test Concept: The IUT is made to send a frame such that the NPDU size is less than or equal to 501 octets (a non-COBS encoded frame) and the frame type is checked. The IUT is then made to send a frame such that the NPDU size is greater than 501 octets (a COBS encoded frame) and the frame type is checked.

It is expected that this test can be executed using ReadPropertyMultiple service requests to generate responses from the IUT of different sizes. If the IUT does not support execution of ReadPropertyMultiple service requests and the IUT cannot be made to send a frame larger than 501 octets by any other means, this test shall be skipped.

Test Steps:

1. MAKE (the IUT generate a frame with an NPDU less than 501 octets)
2. VERIFY (Frame type = 4 | 5)
3. MAKE (the IUT generate a frame with an NPDU greater than 501 octets)
4. VERIFY (Frame type = 32 | 33)

12.1.3.X2 Executing COBS Encoded Frames

Purpose: To verify that the IUT can properly execute COBS encoded frames

Test Concept: A COBS encoded service request is sent to the IUT and proper execution of the request is verified.

It is expected that this test can be executed for server devices using a large ReadPropertyMultiple service request that the server can execute. If the IUT does not support execution of ReadPropertyMultiple service requests, the vendor must provide instructions and means for verifying correct execution of a request.

Test Steps:

1. IF (the IUT supports execution of ReadPropertyMultiple service requests)
   READ V = (Object1), P1
   TRANSMIT (a ReadPropertyMultiple request with an NPDU larger than 501 octets including (Object1), P1)
   VERIFY (Object1),P1 = V
2. ELSE
   (Use vendor supplied instructions to verify execution of a service request)

[In BTL Test Plan, section 10.1.2, replace 135.1 test with BTL version of 10.2.3.2]

10.1.2 Routes Packets between Physical BACnet LANs

…		
~~135.1-2013~~*BTL* - 10.2.3.2 - Route Message from a Local Device to a Local Device		
	Test Conditionality	Must be executed.
	Test Directives	Repeat for each LAN type supported by the device. (The test checks communication in both directions.) One of the tests 10.2.3.2 and 10.2.3.3 must be run by sending the message using a Local Broadcast destination, and the other test must be run by sending the message directly to the IUT's MAC address. This test may be varied by sending a BACnet-ComplexACK-PDU or a BACnet-SimpleACK-PDU instead of a BACnet-Confirmed-Request-PDU; check the value of the data_expecting_reply bit.
	Testing Hints	
…		

[In BTL Specified Tests, add new test that is derived from 135.1-2013 with the changes below]

10.2.3.2 Route Message from a Local Device to a Local Device

Purpose: To verify that the IUT can route a *maximum sized NPDU* unicast message from a local device on Network 1 to a device on Network 2.

Test Concept: A message is sent from Network 1 destined for Network 2 via the IUT and router functionality is verified by observing the message on Network 2. The sequence is repeated in the opposite direction to verify the IUT can also route messages from Network 2 to Network 1. The messages may have an NPDU length, L, such that L equals the Maximum NPDU length as defined in Table 6-1 for the smaller data link.

Test Steps:

1. TRANSMIT PORT A,
   DA = IUT,
   SA = D1A,
   DNET = 2,
   DADR = D2C,
   Hop Count = 255,
   BACnet-Confirmed-Request-PDU,
   'Service Choice' = ~~ReadProperty-Request~~ *WriteProperty-Request,*

'Object Identifier' = (*O1 =* any object identifier),
'Property Identifier' = (*P1 =* any property of the specified object *with a CharacterString datatype),*
'Property Value' = (V = CharacterString with a length such that the NPDU length = L)

2. RECEIVE PORT B,
   DA = D2C,
   SA = IUT,
   SNET = 1,
   SADR = D1A,
   BACnet-Confirmed-Request-PDU,
   'Service Choice' = ~~ReadProperty-Request~~ *WriteProperty-Request,*
   'Object Identifier' = *O1*~~(any object identifier)~~,
   'Property Identifier' = *P1* ~~(any property of the specified object)~~,
   'Property Value' = V
3. TRANSMIT PORT B,
   DA = IUT,
   SA = D2C,
   DNET = 1,
   DADR = D1A,
   Hop Count = 255,
   BACnet-Confirmed-Request-PDU,
   'Service Choice' = ~~ReadProperty-Request~~ *WriteProperty-Request,*
   'Object Identifier' = (*O1 =* any object identifier),
   'Property Identifier' = (*P1 =* any property of the specified object *with a CharacterString datatype),*
   'Property Value' = (V = CharacterString with a length such that the NPDU length = L)
4. RECEIVE PORT A,
   DA = D1A,
   SA = IUT,
   SNET = 2,
   SADR = D2C,
   BACnet-Confirmed-Request-PDU,
   'Service Choice' = ~~ReadProperty-Request~~ *WriteProperty-Request,*
   'Object Identifier' = *O1*~~(any object identifier)~~,
   'Property Identifier' = *P1* ~~(any property of the specified object)~~,
   'Property Value' = V

BTL-15.2s-3: Reduce COVP Test Iterations [BTLWG-658]

Overview:

The COVP changes in 15.2j call for exhaustive testing of COVP. Since the test selection is not automatable (this information is not included in the EPICS), and the changing of the properties is automatable (not all properties will be writable), the effort required to meet this requirement outweighs any benefit had by exhaustive testing.

Changes:

[In BTL Test Plan (with Add 15.2j-1 applied) - modify the Test Directives for all applications of BTL - 9.11.1.X11 and BTL - 9.11.1.X12]

BTL - 9.11.1.X11 - Confirmed Change of Value Notification from Property Value		
	Test Conditionality	Must be executed.
	Test Directives	~~Apply the test to an object and property which supports SubscribeCOVProperty.~~ ~~*Repeat test for at least one object of each type that has at least one property which supports SubscribeCOVProperty, repeat test for all Properties which support SubscribeCOVProperty.*~~ *Apply to at least 1 property of the specified datatype.* *Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVP on one or more of its properties.*
	Testing Hints	
BTL - 9.11.1.X12 - Unconfirmed Change of Value Notification from Property Value		
	Test Conditionality	~~Must be executed.~~
	Test Directives	~~Apply the test to an object and property which supports SubscribeCOVProperty.~~ ~~*Repeat test for at least one object of each type that has at least one property which supports SubscribeCOVProperty, repeat test for all Properties which support SubscribeCOVProperty*~~ *Apply to at least 1 property of the specified datatype.* *Ensure that after all applications of this test (regardless of the property datatype it is applied for), that the test has been applied at least once to each object type which supports COVP on one or more of its properties.*
	Testing Hints	