BACnet® TESTING LABORATORIES ADDENDA

Addendum fix1 to BTL Test Package 26.1

Revision v4
Revised 4/12/2026

Approved by the BTL Working Group on March 5, 2026;
Approved by the BTL Working Group Voting Members on April 9, 2026;
Published on April 14, 2026.

BACnet is a registered trademark of ASHRAE. ASHRAE does not endorse, approve or test products for compliance with ASHRAE standards. Compliance of listed products to the requirements of ASHRAE Standard 135 is the responsibility of BACnet International. BTL is a registered trademark of BACnet International.

[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.

BTL-26.1 fix1-1:7.3.2.39.4 Lighting Command Operation FADE_TO Test [BTLWG-1657] 2
BTL-26.1 fix1-2: 7.3.2.41.3 Denied Access Test and 7.3.2.41.7 Lockout State Test [BTLWG-1661] 4
BTL-26.1 fix1-3: 8.4.2 CHANGE_OF_STATE Tests (ConfirmedEventNotification) Fix [BTLWG-1664] 7
BTL-26.1 fix1-4: Add Purpose to Record_Count_Test [BTLWG-1669] 11
BTL-26.1 fix1-5: Test Plan 10.8.1, Incorrect String Reference [BTLWG-1796] 12
BTL-26.1 fix1-6: RESTART_AUTONEGOTIATION Test Fix [BTLWG-1821] 13
BTL-26.1 fix1-7: Test Concept Fix for Configurable MAC Address Test [BTLWG-1799] 14
BTL-26.1 fix1-8: Fix Routers Execute What-Is-Network Test [BTLWG-1807] 15

In the following document, language to be added to existing clauses within the BTL Test Package 26.1 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.

© 2026 by BACnet International. All rights reserved.

BTL-26.1 fix1-1:7.3.2.39.4 Lighting Command Operation FADE_TO Test [BTLWG-1657]

Overview:

In this test, the "fade-to" operation is examined. In the first half of the test, an ascending check is performed from 0 to V1, and in the second half, a descending check is performed from 100 to V2.

In Step 25, it is checked whether, after approximately half of the "Fading-Time" the Tracking_Value has also covered half the distance from 100 to V2. However, the test specification states exactly the same as in Step 11, namely: VERIFY Tracking_Value ~= V1 / 2.

The correct check should be: VERIFY Tracking_Value ~= (V2 + 100) / 2

Changes:

Checklist Changes

None

Test Plan Changes

[Change all references of 135.1-2025 - 7.3.2.39.4 to BTL]

Specified Test Changes

7.3.2.39.4 Lighting Command Operation FADE_TO Test

Reason for change: wrong requested value for Tracking_Value in Step 25

Purpose: To verify the correct operation of FADE_TO lighting command by observing the value of Present_Value, In_Progress and Tracking_Value.

Test Concept: The TD writes to the Present_Value at each end of the range (i.e., 0% or 100%), and then writes to the Lighting Command Operation with FADE_TO with a long enough fade-time to allow In_Progress and Tracking_Value to be observed while set to FADE_ACTIVE. The Tracking_Value will be checked at the end of the fade to verify that it tracked the target level. The IUT shall be tested for fade up (0% to 100%) and fade down (100% to 0%).

Configuration Requirements: O1 shall be configured such that all slots in the Priority_Array numerically less than PTY1 have a value of NULL and no internal algorithms are issuing commands to O1 at a priority numerically less than or equal to PTY1. V1 > 1 and V2 < 100%

Test Steps:

-- Start with 0% Present_Value to test fade up
1. WRITE Present_Value = 0, ARRAY INDEX = PTY1
2. VERIFY Present_Value = 0
3. WAIT **Internal Processing Fail Time**
4. VERIFY Tracking_Value = 0

-- Write a FADE_TO command (operation, target-level, priority, fade-time)
5. WRITE Lighting_Command = (FADE_TO, V1, PTY1, FT)
6. WAIT **Internal Processing Fail Time**
7. VERIFY Priority_ Array = V1, ARRAY INDEX = PTY1
8. VERIFY Present_Value = V1

-- In a half way of fading up, check In_Progress and Tracking_Value

© 2026 by BACnet International. All rights reserved.

9. WAIT FT/2
10. VERIFY In_Progress = FADE_ACTIVE,
11. VERIFY Tracking_Value ~= V1 / 2
12. WAIT FT/2

-- When fading up is completed, check In_Progress and Tracking_Value
13. VERIFY In_Progress = IDLE
14. VERIFY Tracking_Value = V1

-- Now repeat the test with 100% Present_Value to test fade down
15. WRITE Present_Value = 100, ARRAY INDEX = PTY1
16. VERIFY Present_Value = 100
17. WAIT **Internal Processing Fail Time**
18. VERIFY Tracking_Value = 100

-- Write a FADE_TO command (operation, target-level, priority, fade-time)
19. WRITE Lighting_Command = (FADE_TO, V2, PTY1, FT)
20. WAIT **Internal Processing Fail Time**
21. VERIFY Priority_ Array = V2, ARRAY INDEX = PTY1
22. VERIFY Present_Value = V2

-- In a half way of fading down, check In_Progress and Tracking_Value
23. WAIT FT/2
24. VERIFY In_Progress = FADE_ACTIVE,
25. VERIFY Tracking_Value ~= ~~*V1 / 2*~~ *(V2+100)/2*
26. WAIT FT/2

-- When fading down is completed, check In_Progress and Tracking_Value
27. VERIFY In_Progress = IDLE
28. VERIFY Tracking_Value = V2

© 2026 by BACnet International. All rights reserved.

BTL-26.1 fix1-2: 7.3.2.41.3 Denied Access Test and 7.3.2.41.7 Lockout State Test [BTLWG-1661]

Overview:

Tests "7.3.2.41.3 Denied Access Test" and "7.3.2.41.7 Lockout State Test" are missing the object type in the value for Access_Event_Credential which is requested to be verified.

Changes:

Checklist Changes

None

Test Plan Changes

[Change all references for test 7.3.2.41.3 and test 7.3.2.41.7 from 135.1-2025 to BTL]

Specified Test Changes

7.3.2.41.3 Denied Access Test

Reason for change: missing object type in object identifier.

Purpose: To verify that a credential that is not allowed access to this access point at this time is denied access. There are a number of reasons why a credential may be denied access and this test tests the situations which must be supported by the access point.

Test Concept: To test that a credential, which is not allowed access to this access point, is presented at the access point with the result that access is allowed and the appropriate access event is generated.

Configuration Requirements: See Clause 7.3.2.41. This test requires the following additional configuration: The vendor shall provide a set of credentials which correspond to Access Credential objects configured such that access to the access point shall be denied for the following reasons:

a. DENIED_POINT_NO_ACCESS_RIGHTS = C1
b. DENIED_NO_ACCESS_RIGHTS = C2
c. DENIED_ZONE_NO_ACCESS_RIGHTS = C3
d. DENIED_CREDENTIAL_NOT_YET_ACTIVE = C4
e. DENIED_CREDENTIAL_EXPIRED = C5
f. DENIED_CREDENTIAL_MANUAL_DISABLE = C6
g. DENIED_INCORRECT_AUTHENTICATION_FACTOR = C7
h. DENIED_OUT_OF_TIME_RANGE = C8
i. DENIED_THREAT_LEVEL = C9
j. DENIED_PASSBACK = C10
k. DENIED_UNEXPECTED_LOCATION_USAGE = C11
l. DENIED_MAX_ATTEMPTS = C12
m. DENIED_AUTHENTICATION_FACTOR_LOST = C13
n. DENIED_AUTHENTICATION_FACTOR_STOLEN= C14
o. DENIED_AUTHENTICATION_FACTOR_DAMAGED = C15
p. DENIED_AUTHENTICATION_FACTOR_DESTROYED = C16
q. DENIED_AUTHENTICATION_FACTOR_DISABLED = C17
r. DENIED_AUTHENTICATION_FACTOR_ERROR = C18
s. DENIED_CREDENTIAL_UNASSIGNED = C19
t. DENIED_CREDENTIAL_NOT_PROVISONED = C20
u. DENIED_CREDENTIAL_LOCKOUT = C21
v. DENIED_CREDENTIAL_MAX_DAYS = C22
w. DENIED_CREDENTIAL_MAX_USES = C23

© 2026 by BACnet International. All rights reserved.

x. DENIED_CREDENTIAL_DISABLED = C24
y. DENIED_LOCKOUT = C25

Notes to Tester: If the IUT does not support any of the above denial reasons, then the corresponding credentials are not required to be supplied.

Test Steps:

1. REPEAT C = (C1…C25) DO {
   READ EventTag = Access_Event_Tag
   MAKE (present the credential corresponding to C at the credential reader for this access point)
   VERIFY Access_Event = (denied reason corresponding to credential C)
   VERIFY Access_Event_Time = (the time that credential C was presented)
   VERIFY Access_Event_Credential = C
   VERIFY Access_Event_Tag = EventTag + 1
   }

-- verify unknown credential event
2. READ EventTag = Access_Event_Tag
3. MAKE (present a credential which does not correspond to any configured Access Credential object at the credential reader for this access point)
4. VERIFY Access_Event = DENIED_UNKNOWN_CREDENTIAL
5. VERIFY Access_Event_Time = (the time that the credential was presented)
6. VERIFY (Access_Event_Credential = (*8,* 4194303, ?, 4194303)
7. VERIFY Access_Event_Tag = EventTag + 1

7.3.2.41.7 Lockout State Test

Reason for change: missing object type in object identifier.

Purpose: To verify that access is denied for any credential when the access point is in the lockout state. To verify that using an invalid credential at the access point multiple times will cause the access point to go into a lockout state. To verify that the lockout will automatically relinquish after the specified time.

Test Concept: A credential which will result in denied access is repeatedly presented at the access point until the access point becomes locked out. When the access point becomes locked, valid credentials will also be denied access until the lockout relinquish time has expired.

Configuration Requirements: See Clause 7.3.2.41. This test requires the following additional configuration:
a) The Max_Failed_Attempts property, if present, has a value greater than 0.
b) An active credential with valid access rights for the access point shall be represented by Access Credential object C1.
c) An active credential with no valid access rights for the access point shall be represented by Access Credential object C2.
d) The Failed_Attempts_Events list, if present, shall have at least one entry corresponding to the reason why C2 is denied access.
e) The Lockout_Relinquish_Time has a value greater than 0.

Test Steps:

-- verify that valid credentials are denied when the Lockout property is TRUE
1. WRITE Lockout = TRUE
2. WAIT Internal Processing Fail Time
3. VERIFY Access_Event = LOCKOUT_OTHER
4. VERIFY Access_Event_Time = (the time that TRUE was written to the Lockout property)
5. VERIFY Access_Event_Credential = (*8,*4194303, ?, 4194303)
6. MAKE (present credential C1 at credential reader for this access point)
7. VERIFY Access_Event = DENIED_LOCKOUT
8. VERIFY Access_Event_Time = (the time that credential C1 was presented)

© 2026 by BACnet International. All rights reserved.

9. VERIFY Access_Event_Credential = C1

-- verify that using an invalid credential at the an access point multiple times will cause the access point to go into a lockout state
10. WRITE Lockout = FALSE
11. WAIT Internal Processing Fail Time
12. VERIFY Access_Event = LOCKOUT_RELINQUISHED
13. VERIFY Access_Event_Time = (the time that FALSE was written to the Lockout property)
14. VERIFY Access_Event_Credential = (*8,*4194303, ?, 4194303)
15. IF (Failed_Attempts and Max_Failed_Attempts are supported) THEN
        REPEAT X= (1 to Max_Failed_Attempts + 1) DO {
            READ FailedAttempts = Failed_Attempts
            MAKE (present credential C2 at credential reader for this access point)
            VERIFY (Failed_Attempts = FailedAttempts + 1)
        }
16. VERIFY (Lockout = TRUE)
17. VERIFY (Access_Event = LOCKOUT_MAX_ATTEMPTS)
18. VERIFY (Access_Event_Time = the time that Lockout was set to TRUE)
19. VERIFY (Access_Event_Credential = C2)
20. MAKE (present credential C1 at credential reader for this access point)
21. VERIFY (Access_Event = DENIED_LOCKOUT)
22. VERIFY (Access_Event_Time = the time that credential C1 was presented)
23. VERIFY (Access_Event_Credential = C1)

-- verify that the lockout will automatically relinquish after the specified time
24. WAIT Lockout_Relinquish_Time
25. VERIFY (Lockout = FALSE)
26. VERIFY (Access_Event = LOCKOUT_RELINQUISHED)
27. VERIFY (Access_Event_Time = the time that Lockout was set to FALSE)
28. VERIFY Access_Event_Credential = (*8,*4194303, ?, 4194303)
29. MAKE (present credential C1 at credential reader for this access point)
30. VERIFY (Access_Event = GRANTED)
31. VERIFY (Access_Event_Time = the time that credential C1 was presented)
32. VERIFY (Access_Event_Credential = C1)

© 2026 by BACnet International. All rights reserved.

BTL-26.1 fix1-3: 8.4.2 CHANGE_OF_STATE Tests (ConfirmedEventNotification) Fix [BTLWG-1664]

Overview:

The Test 8.4.2 in ANSI/ASHRAE Standard 135.1-2023 does not include the optional parameter “pTimeDelayNormal” for the case it exits.
Based on ANSI/ASHRAE Standard 135-2020 – 13.3.2 CHANGE_OF_STATE Event Algorithm

pTimeDelayNormal: This parameter, of type Unsigned, represents the time, in seconds, that the Normal conditions must exist before a NORMAL event state is indicated. If no value is available for this parameter, then it takes on the value of the pTimeDelay parameter.

pTimeDelayNormal should be included.

Changes:

Checklist Changes

None

Test Plan Changes

[Move from 135.1 to BTL Specified tests for all instances of 8.4.2]

Specified Test Changes

8.4.2 CHANGE_OF_STATE Tests (ConfirmedEventNotification)

Reason for change: Optional Parameter “pTimeDelayNormal” has not been included in the Test – Step 2.

Purpose: To verify the correct operation of the CHANGE_OF_STATE event algorithm.

Test Concept: The object begins the test in a NORMAL state. The Present_Value (referenced property) is changed to a value that is one of the values designated in List_Of_Values. After the time delay expires the object should enter the OFFNORMAL state and transmit an event notification message. The Present_Value (referenced property) is then changed to a value corresponding to a NORMAL state. After the time delay the object should enter the NORMAL state and transmit an event notification message. If the IUT claims conformance to Protocol_Revision 12 or lower, and a Multi-state Input or Multistate Value object is being tested, the transition to and from the FAULT state is also tested.

Configuration Requirements: The IUT shall be configured such that the Event_Enable property has a value of TRUE for the TO-OFFNORMAL, TO-FAULT and TO-NORMAL transitions. The 'Issue_Confirmed_Notifications' parameter shall have a value of TRUE. The event-generating objects shall be in a NORMAL state at the start of the test.

If the IUT claims conformance to Protocol_Revision 12 or lower, and supports intrinsic reporting for Multi-state Input or Multi-state Value objects, the intrinsic reporting object shall be configured with at least one of the two properties, Alarm_Values (referred to as pAlarmValues in the test steps) and Fault_Values (referred to as pFaultValues in the test steps), containing at least one value.

If the IUT claims conformance to Protocol_Revision 12 or lower, and supports intrinsic reporting for Binary Input or Binary Value objects, the intrinsic reporting object shall be configured with the Alarm_Value property (referred to as pAlarmValues in the test steps) containing at least one value.

If the IUT claims conformance to Protocol_Revision 12 or lower, and supports algorithmic change reporting with an Event_Type of CHANGE_OF_STATE, the List_Of_Values parameter of the Event_Parameters property (referred to as pAlarmValues in the test steps) shall contain at least one value.

© 2026 by BACnet International. All rights reserved.

If the IUT claims conformance to Protocol_Revision 13 or greater, and supports the CHANGE_OF_STATE algorithm, the IUT shall be configured with at least one value for pAlarmValues.

Test Steps:

1. VERIFY pCurrentState = NORMAL
2. IF ((Protocol_Revision is present AND Protocol_Revision >= 13)
   OR ((Protocol_Revision is present AND Protocol_Revision < 13)
   AND (pAlarmValues contains at least one value))) THEN {
   IF (pMonitoredValue is writable) THEN
   WRITE pMonitoredValue = (a value from pAlarmValues)
   ELSE
   MAKE (pMonitoredValue have a value pAlarmValues)
   WAIT (pTimeDelay)
   BEFORE **Notification Fail Time**
   RECEIVE ConfirmedEventNotification-Request,
   'Process Identifier' = (any valid process ID),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (the intrinsic reporting object being tested or the Event Enrollment object being tested),
   'Time Stamp' = (T1, any valid time stamp),
   'Notification Class' = (the configured notification class),
   'Priority' = (the value configured to correspond to a TO_OFFNORMAL transition),
   'Event Type' = CHANGE_OF_STATE,
   'Message Text' = (optional, any valid message text),
   'Notify Type' = EVENT | ALARM,
   'AckRequired' = TRUE | FALSE,
   'From State' = NORMAL,
   'To State' = OFFNORMAL,
   'Event Values' = pMonitoredValue, pStatusFlags
   TRANSMIT BACnet-SimpleACK-PDU
   IF (Protocol_Revision is present AND Protocol_Revision >= 13) THEN
   VERIFY Status_Flags = (TRUE, FALSE, ?, ?)
   VERIFY pCurrentState = OFFNORMAL
   IF (Protocol_Revision is present AND Protocol_Revision >= 1) THEN
   VERIFY Event_Time_Stamps = (T1, Ta, Tb)
   IF (pMonitoredValue is writable) THEN
   WRITE pMonitoredValue = (a value that corresponds to a NORMAL state)
   ELSE
   MAKE (pMonitoredValue have a value that corresponds to a NORMAL state)
   WAIT ~~(pTimeDelay)~~ *(pTimeDelayNormal)*
   BEFORE Notification Fail Time
   RECEIVE ConfirmedEventNotification-Request,
   'Process Identifier' = (any valid process ID),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (the intrinsic reporting object being tested or the Event Enrollment object being tested),
   'Time Stamp' = (T2, any valid time stamp),
   'Notification Class' = (the configured notification class),
   'Priority' = (the value configured to correspond to a TO_NORMAL transition),
   'Event Type' = CHANGE_OF_STATE,
   'Message Text' = (optional, any valid message text),
   'Notify Type' = EVENT | ALARM,
   'AckRequired' = TRUE | FALSE,
   'From State' = OFFNORMAL,
   'To State' = NORMAL,
   'Event Values' = pMonitoredValue, pStatusFlags
   TRANSMIT BACnet-SimpleACK-PDU

© 2026 by BACnet International. All rights reserved.

```
        IF (Protocol_Revision is present AND Protocol_Revision >= 13) THEN
            VERIFY Status_Flags = (FALSE, FALSE, ?, ?)
        VERIFY pCurrentState = NORMAL
        IF (Protocol_Revision is present AND Protocol_Revision >= 1) THEN
            VERIFY Event_Time_Stamps = (T1, Ta, T2)
    }
3.  IF ((Protocol_Revision is present AND Protocol_Revision < 13)
            AND (intrinsic reporting is being tested)
            AND (the intrinsic reporting object is configured with pFaultValues containing at least one values)) THEN {
        IF (pMonitoredValue is writable) THEN
            WRITE pMonitoredValue = (a value from pFaultValues)
        ELSE
            MAKE (pMonitoredValue have a value from pFaultValues)
        BEFORE Notification Fail Time
            RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' = (any valid process ID),
                'Initiating Device Identifier' = IUT,
                'Event Object Identifier' = (the intrinsic reporting object being tested),
                'Time Stamp' = (Tfault: any valid timestamp),
                'Notification Class' = (the configured notification class),
                'Priority' = (the value configured to correspond to a TO_FAULT transition),
                'Event Type' = CHANGE_OF_STATE,
                'Message Text' = (optional, any valid message text),
                'Notify Type' = EVENT | ALARM,
                'AckRequired' = TRUE | FALSE,
                'From State' = NORMAL,
                'To State' = FAULT,
                'Event Values' = pMonitoredValue, pStatusFlags
        TRANSMIT BACnet-SimpleACK-PDU
        VERIFY pCurrentState = FAULT
        IF (Protocol_Revision is present AND Protocol_Revision >= 1) THEN
            VERIFY Event_Time_Stamps = (Toffnormal, Tfault, Tnormal)
        VERIFY pCurrentReliability = MULTI_STATE_FAULT
        IF (pMonitoredValue is writable) THEN
            WRITE pMonitoredValue = (a value that corresponds to a NORMAL state)
        ELSE
            MAKE (pMonitoredValue have a value that corresponds to a NORMAL state)
        BEFORE Notification Fail Time
            RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' = (any valid process ID),
                'Initiating Device Identifier' = IUT,
                'Event Object Identifier' = (the intrinsic reporting object being tested),
                'Time Stamp' = (Tnormal: any valid timestamp),
                'Notification Class' = (the configured notification class),
                'Priority' = (the value configured to correspond to a TO_NORMAL transition),
                'Event Type' = CHANGE_OF_STATE,
                'Message Text' = (optional, any valid message text),
                'Notify Type' = EVENT | ALARM,
                'AckRequired' = TRUE | FALSE,
                'From State' = FAULT,
                'To State' = NORMAL,
                'Event Values' = pMonitoredValue, pStatusFlags
        TRANSMIT BACnet-SimpleACK-PDU
        VERIFY pCurrentState = NORMAL
        IF (Protocol_Revision is present AND Protocol_Revision >= 1) THEN
            VERIFY Event_Time_Stamps = (Toffnormal, Tfault, Tnormal)
    }
```

Notes to Tester: The time stamps indicated by "Ta" and "Tb" can have a value that indicates an unspecified time or a time that precedes the timestamp T1.

© 2026 by BACnet International. All rights reserved.

© 2026 by BACnet International. All rights reserved.

BTL-26.1 fix1-4: Add Purpose to Record_Count_Test [BTLWG-1669]

Overview:

Test 7.3.2.24.8 Record_Count Test does not include a Purpose statement. This work item adds it back.

Changes:

Checklist Changes

None

Test Plan Changes

[Change all references to 135.1-7.3.2.24.8 Record_Count Test to BTL]

Specified Test Changes

7.3.2.24.8 Record_Count Test
Reason for Change: To add the Purpose statement back to the test.

Purpose: To verify that the Record_Count property indicates the number of records that are stored in the Log_Buffer.

Test Concept: The logging object is configured to acquire data by whatever means. Record_Count is set to zero and Log_Buffer is read to verify that only one record is present and that it is the buffer-purged event. Collection of data proceeds until Record_Count is about Buffer_Size/2, collection is halted and Log_Buffer is read to verify the Record_Count value. Collection then resumes until Buffer_Size records are read; collection is then halted and Log_Buffer read to verify the Record_Count again.

Configuration Requirements: Start_Time, if present, shall be configured with a date and time preceding the beginning of the test. Stop_Time, if present shall be configured with the latest possible date and time, in order that it occurs after the end of the test. Enable shall be set to FALSE.

Test Steps:

1. WRITE Record_Count = 0
2. WAIT **Internal Processing Fail Time**
3. CHECK (Log_Buffer contains one entry, and it is the buffer-purged event)
4. WRITE Enable = TRUE
5. WHILE (Record_Count < Buffer_Size/2) DO { }
6. WRITE Enable = FALSE
7. WAIT **Internal Processing Fail Time**
8. CHECK (that Log_Buffer has the number of records indicated by Record_Count)
9. WRITE Enable = TRUE
10. WHILE (Record_Count < Buffer_Size) DO { }
11. WRITE Enable = FALSE
12. WAIT **Internal Processing Fail Time**
13. CHECK (that Log_Buffer has the number of records indicated by Record_Count)

© 2026 by BACnet International. All rights reserved.

BTL-26.1 fix1-5: Test Plan 10.8.1, Incorrect String Reference [BTLWG-1796]

Overview:

The BTL Test Plan section ‘10.8.1 Base Requirements’ under NM-FDR-A refers to ‘Is able to operate as a Foreign Device’ but this doesn’t match the wording used in the BTL checklist.

Changes:

Checklist Changes

None

Test Plan Changes

10.8 Network Management – Foreign Device Registration - A

[Modify Test Directives of Verify Checklist item under 10.8.1 of BTL Test Plan]

10.8.1 Base Requirements

Base requirements must be met by any IUT claiming conformance to this BIBB.

Verify Checklist		
	Test Conditionality	Must be executed.
	Test Directives	Verify that the IUT claims support for Data Link Layer and 'Is able to operate ~~as a Foreign Device’~~*.in Foreign Mode'*.
	Testing Hints	

Specified Test Changes

None

© 2026 by BACnet International. All rights reserved.

BTL-26.1 fix1-6: RESTART_AUTONEGOTIATION Test Fix [BTLWG-1821]

Overview:

Link_Speed_Autonegotiate should be FALSE.

Changes:

Checklist Changes

None

Test Plan Changes

None

Specified Test Changes

[Update BTL Specified Test 7.3.2.46.3.6.2]

7.3.2.46.3.6.2 RESTART_AUTONEGOTIATION Command Failure Test

Reason for Change: Updated to make the Link_Speed_Autonegotiate check optional. Fix Link_Speed_Autonegotiate.

Purpose: To verify that Network Port objects respond to the RESTART_AUTONEGOTIATION command with the correct error codes when the command is not supported / enabled.

Test Concept: Starting with a Network Port object which is not configured to auto-negotiate its link speed or which does not support the RESTART_AUTONEGOTIATION, command it to restart auto-negotiation. Verify that the correct error code is returned.

Configuration Requirements: If the network port support auto-negotiation, disable it. If the IUT does not support the Command property, or all Network Port object support auto-negotiation and it cannot be disabled, then this test shall be skipped.

Test Steps:
-- make sure our initial conditions are good
1. *IF Link_Speed_Autonegotiate is present THEN*
~~VERIFY Link_Speed_Auto-negotiate = TRUE~~
2. VERIFY Link_Speed_Autonegotiate = ~~TRUE~~FALSE

-- request the renewal, and wait for it to timeout
~~2~~*3*. TRANSMIT WriteProperty-Request,
'Object Identifier' = (the Network Port object),
'Property Identifier' = Command,
'Property Value' = RESTART_AUTONEGOTIATION

~~3~~*4*. IF the port does not support auto-negotiation THEN
5. RECEIVE BACnet-Error-PDU
'Error Class' = PROPERTY,
'Error Code' = OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED
ELSE
6. RECEIVE BACnet-Error-PDU
'Error Class' = PROPERTY,
'Error Code' = VALUE_OUT_OF_RANGE

© 2026 by BACnet International. All rights reserved.

BTL-26.1 fix1-7: Test Concept Fix for Configurable MAC Address Test [BTLWG-1799]

Overview:

The Test Concept does not clearly describe the device receiving the request, confusing the reader.

Changes:

Checklist Changes

None

Test Plan Changes

[In BTL Test Plan, change all occurrences of test 9.44.1.2 from 135.1-2025 to BTL]

Specified Test Changes

[Move test 9.44.1.2 from 135.1 into BTL Specified Tests and change as noted below]

9.44.1.2 Configurable MAC Address

Reason for Change: Fixed the Test Concept.

Purpose: To verify the IUT can be configured with a MAC address using the You-Are service.

Test Concept: TD sends a You-Are request to configure *IUT's* ~~TD's~~ MAC address.

Configuration Requirements: The IUT is not configured with a MAC address. The IUT is configured with a Device object instance number (X). If the IUT cannot be configured in this way, this test shall be skipped.

Notes to Tester: The destination address used by TD shall be selected such that the IUT will receive the messages.

Test Steps:
1. TRANSMIT
   DESTINATION = LOCAL BROADCAST | GLOBAL BROADCAST | REMOTE BROADCAST,
   You-Are Request,
   'Vendor Identifier' = (the IUT's Vendor_Identifier),
   'Model Name' = (the IUT's Model_Name),
   'Serial Number' = (the IUT's Serial_Number),
   'Device Identifier' = (absent),
   'Device MAC Address' = (a valid MAC address)
2. IF (the IUT is not an MS/TP subordinate node)
   BEFORE **Internal Processing Fail Time**
   RECEIVE
   DESTINATION = TD | LOCAL BROADCAST | GLOBAL BROADCAST | REMOTE BROADCAST,
   I-Am Request,
   'Device Identifier' = (Device, X),
   'Max APDU Length Accepted' = (any valid value),
   'Segmentation Supported' = (any valid value),
   'Vendor Identifier' = (the IUT's Vendor_Identifier)

© 2026 by BACnet International. All rights reserved.

BTL-26.1 fix1-8: Fix Routers Execute What-Is-Network Test [BTLWG-1807]

Overview:

Test ‘135.1-2025-10.2.8 Routers Execute What-Is-Network-Number’ is for routers, and it allows the router to wait up to 10 seconds to respond to a What-Is-Network-Number request. However, 135-2024 section '6.4.19 What-Is-Network-Number' only says that non-routers may wait up to 10 seconds.

Changes:

Checklist Changes

None

Test Plan Changes

[Change all references of test 10.2.8 from 135.1-2025 to BTL Specified Test]

Specified Test Changes

[Move test 10.2.8 from 135.1 into BTL Specified Tests and modify as noted below]

10.2.8 Routers Execute What-Is-Network-Number

Reason for Change: Remove 10 second timer which is only applicable for non-routers, and update test step to use N1.

Purpose: To verify that a router responds to a What-Is-Network-Number request ~~within 10 seconds~~.

Test Concept: A What-Is-Network-Number is broadcast on the local network and the tester verifies that the IUT responds with a Network-Number-Is message ~~within 10 seconds~~.

Configuration Requirements: The IUT knows its network number, N1. If the IUT claims a protocol revision of less than 11, this test shall be skipped.

Test Steps:

1. TRANSMIT What-Is-Network-Number,
   DESTINATION = LOCAL_BROADCAST
2. *BEFORE*~~Before 10s +~~ Internal Processing Fail Time
3. RECEIVE Network-Number-Is,
   Network Number = *N1*~~(the configured value)~~,
   Configured = (any valid value)

© 2026 by BACnet International. All rights reserved.