

BACnet® TESTING LABORATORIES

MANUAL TEST INSTRUCTIONS

Revision 27
Revised November 1, 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.

Table of Contents

1 Purpose 2
2 Manual Tests 3
2.1 The General Manual Test Method 3
2.1.1 Purpose 3
2.1.2 Tools Required 3
2.2 Test Step Constructs 3
2.2.1 IF/THEN/ELSE 3
2.2.2 REPEAT 4
2.2.3 ERROR 4
2.2.4 CHECK 4
2.2.5 MAKE 5
2.2.6 TRANSMIT 5
2.2.7 RECEIVE 6
2.2.8 WAIT 6
2.2.9 WRITE 7
2.2.10 VERIFY 7
2.2.11 BEFORE 7
2.2.12 WHILE 8
2.3 Verifying Received Packets 9
3 Manual Tests 9
3.1 EPICS Consistency Tests 9
3.2 135.1 - 7.1 Read Support for Properties in the Test Database 9
3.3 135.1 - 7.2.2 Write Support Test Procedure 9

© 2026 by BACnet International. All rights reserved.

1 Purpose

This document describes how to manually apply the BTL tests, for situations where scripting and other automated procedures do not apply or cannot be applied

Most of the tests found in 135.1 can be applied using a general manual test method as described in section 2.1.

© 2026 by BACnet International. All rights reserved.

2 Manual Tests

2.1 The General Manual Test Method

2.1.1 Purpose

This manual test method is used to apply tests defined using the TCSL syntax, defined in clause 4 of ***ASHRAE 135.1***, such as those tests defined in ***ASHRAE 135.1*** and the ***BTL Specified Tests*** document. The method is intended to provide the tester with enough instruction on how to read a test definition and convert the definition into manual test steps.

2.1.2 Tools Required

The list of available tools to perform manual testing is available at https://btl.org/pre-testing-tools/

Although no longer maintained, VTS is still a viable test tool to perform many of the manual tests defined in ***ASHRAE 135.1***and ***BTL Specified Tests***. The tester should have a general understanding of VTS and how to perform the basic testing functions: generating BACnet packets, capturing network traffic, reading captured network traffic, reading and writing BACnet properties in the IUT. For more information on how to perform these functions, refer to the ***VTS Usage Guide***.

Preconditions:

1. Setup VTS as per ***VTS Usage Guide***, ensuring that VTS is communicating with the IUT.
2. Setup the IUT as per the configuration requirements described in the test.
3. For each step in the test, perform the test step as per the guidelines below. If the IUT fails to perform any step, or the ERROR test statement is encountered then the IUT shall fail the test.

2.2 Test Step Constructs

The process for performing a test step starts with understanding the test statement. The following list describes how to apply a test step of each of the given types.

2.2.1 IF/THEN/ELSE

The IF statement allows for conditional test step execution based on the IUT's implementation and its characteristics. The tester must evaluate the IF condition to determine which set of test steps to execute.

2.2.1.1 Execution Steps

1. Evaluate the IF condition. In some cases, the tester will be required to read property values from the IUT in order to evaluate the IF condition. To do so, the tester should initiate a BACnet request using a test tool to determine the property value. If, while evaluating the IF condition, the IUT fails to correctly respond to a BACnet request, then the IUT shall fail the test. The tester shall report the failure in the Results section for the test along with the test step number.
2. If the condition evaluates to TRUE, then the test step immediately following the THEN keyword shall be executed.
3. If the condition evaluates to FALSE and the IF statement contains an optional ELSE portion, the test step following the ELSE shall be executed. If the condition evaluates to FALSE and the optional ELSE portion is absent, the test shall skip the test step immediately following the THEN keyword.

2.2.1.2 Example

In the below example, the tester would use a test tool to change the Present_Value property of the object being tested if the property is writable using BACnet services, otherwise the tester would perform whatever steps are required to change the value of the Present_Value property.

6. IF (Present_Value is writable) THEN
7. WRITE Present_Value = X

ELSE

8. MAKE (Present_Value = X)

© 2026 by BACnet International. All rights reserved.

The following example IF condition requires that the tester read two property values from the IUT.

9. IF (Record_Count = Buffer_Size) THEN

2.2.2 REPEAT

The REPEAT statement allows for test steps to be repeated a specified number of times. The tester must repeat the test steps contained within the REPEAT construct based on the repeat conditions.

2.2.2.1 Execution Steps

1. Evaluate the REPEAT condition. In some cases, the tester will be required to read property values from the IUT in order to evaluate the REPEAT condition. To do so, the tester should initiate a BACnet request using a test tool to determine the property value. If, while evaluating the REPEAT condition, the IUT fails to correctly respond to a BACnet request, then the IUT shall fail the test. The tester shall report the failure in the Results section for the test along with the test step number and the values of any parameters used in that step.
2. If the condition evaluates to TRUE, execute the test steps contained within the REPEAT statement's body. Repeat from step 1.
3. If the condition evaluates to FALSE, skip the test steps contained within the REPEAT statement's body and move on to the next test step.

2.2.2.2 Example

In the below example, the tester would repeat the VERIFY step based on the number of properties included in a previous test step.

3. REPEAT SV = (each value in DateSpecials +TimeSpecials) DO {
4. WRITE P1 = (D1+T1 updated with the value SV)
5. VERIFY P1 = (D1+T1 updated with the value SV)
   }

2.2.3 ERROR

The ERROR statement is used in a test definition when a condition results in the IUT automatically failing of the test. The ERROR statement is usually found within an IF statement.

2.2.3.1 Execution Steps

1. Fail the test

2.2.3.2 Example

ERROR "Incorrect MS/TP Frame Data CRC undetected."

2.2.4 CHECK

The CHECK statement is used in a test definition when the tester must verify that some action by the tester should result in a change to the IUT that is not network visible.

There are a number of tests that use the CHECK statement to ensure that the IUT sends no packets in response to a packet sent by the tester. In these tests, it is expected that the IUT is in a state where it would not normally send any packets unless queried and that there are no other BACnet devices on the network that would interrogate the IUT. The tester should ensure that these conditions exist. If the tester cannot ensure that these conditions exist, then the tester must verify that any packets sent by the IUT were not in response to the tester sent packet.

2.2.4.1 Execution Steps

1. Verify that the IUT exhibited or is exhibiting the desired behaviour.
2. If the IUT does not exhibit the desired behaviour, then the IUT shall fail the test. The tester shall report the failure in the Results section for the test along with the test step number.

© 2026 by BACnet International. All rights reserved.

3. If the IUT does exhibit the desired behaviour, continue with the next test step.

2.2.4.2 Example

In the following example, the tester would check for whatever vendor defined actions the IUT would take when it resets.

11. CHECK (the IUT reset)

In the following example, the test would be checking to ensure that no packets are sent by the IUT. If the IUT did send packets, the tester should ensure that those packets were unrelated to the packet transmitted by the tester.

11. TRANSMIT
    DESTINATION = IUT,
    SOURCE = TD,
    Initialize-Routing-Table
    Number of Ports = 0
12. WAIT **Internal Processing Fail Time**
13. CHECK(that the IUT did not reset and that the IUT did not send any packets)

2.2.5 MAKE

The MAKE statement is used in a test definition when the tester must cause a specific change to the IUT. The tester may, or may not, be able to use a test tool to cause the change.

2.2.5.1 Execution Steps

1. Make the IUT exhibit the behaviour described within the MAKE statement.
2. If the specific change cannot be made to the IUT, then the IUT shall fail the test. The tester shall report the failure in the Results section for the test along with the test step number.

2.2.5.2 Example

In the following example, the tester might use a test tool to change the value of the property. If the IUT cannot be made to take on the value x, yet x is a valid value for the property in question, then the IUT shall fail the test.

11. MAKE (Present_Value have a value x such that x corresponds to a NORMAL state)

2.2.6 TRANSMIT

The TRANSMIT statement is used in a test definition when the tester must cause a specific packet to be sent on the BACnet network.

The TRANSMIT statement describes the packet to send by providing a list of the packet parameters and the values that these are to be set to. Many of the values will be based on the parameters chosen for the test (such as the object or property being tested), or will be based on previous steps in the test.

2.2.6.1 Execution Steps

1. Use a test tool to generate the packet described in the test step. Unless otherwise stated in the test step, the packet being generated is to be sent to the IUT.

2.2.6.2 Example

In the following example, the tester would use a test tool to send a ReadProperty-Request packet to the IUT. Note that the ‘Object Identifier’ parameter shall be set to reference the object that was chosen for this test and documented in the IUT’s Test Plan.

11. TRANSMIT ReadProperty-Request,
    ‘Object Identifier’ = (the object being tested),

© 2026 by BACnet International. All rights reserved.

‘Property Identifier’ = Present_Value

2.2.7 RECEIVE

The RECEIVE statement is used in a test definition when the tester must check that the IUT sent a specific packet. In general, this statement follows a TRANSMIT or MAKE statement and is used to catch the IUT’s response. The tester could verify that the packet was received by examining the Wireshark packet log. Upon finding the packet, the tester must take great care to ensure that ALL of the parameters of the packet are correct. For a detailed description of how to examine packets in the packet log refer to section 2.3, Verifying Received Packets.

In some cases, the RECEIVE statement does not follow a TRANSMIT or MAKE statement. In these cases there is an implied MAKE statement. The tester will be required to understand what is desired and ensure that the IUT is made to perform the action. If the IUT cannot be made to perform the action then the IUT shall fail the test. An example of this can be found in 135.1 in test *8.10.1 Confirmed Notifications Subscription*.

2.2.7.1 Execution Steps

1. Wait for the IUT to generate the desired packet. In general the wait should not be long. The IUT will usually respond before the tester has the chance to check for the response.
2. Check the network packet trace for the response. Verify that each parameter in the response packet sent by the IUT matches the parameter described by the test step.
3. If the IUT does not respond, or any parameter does not match then the IUT shall fail the test. The tester shall report the failure in the Results section for the test along with the test step number.

2.2.7.2 Example

11. TRANSMIT ReadProperty-Request,
    ‘Object Identifier’ = (the object being tested),
    ‘Property Identifier’ = Change_Of_State_Count
12. RECEIVE ReadProperty-ACK,
    ‘Object Identifier’ = (the object being tested),
    ‘Property Identifier’ = Change_Of_State_Count,
    ‘Property Value’ = (any valid value, N)

2.2.8 WAIT

The WAIT statement is used in a test definition to inform the tester to wait for a specified period of time before continuing with the test. The tester may prepare for the next test step but must wait until the specified amount of time has passed before executing it.

2.2.8.1 Execution Steps

1. Stop executing test steps for the specified period.
2. Continue the test with the next test step.

2.2.8.2 Example

In the following example test, the WAIT test step is used to ensure that the minimum on time setup is performing correctly in the IUT. (Hint: while waiting for the Minimum_On_Time to complete in this example, the tester could prepare the ReadProperty packet for the VERIFY statement.)

1. WRITE Present_Value = ACTIVE, PRIORITY = 7
2. VERIFY Present_Value = ACTIVE
3. VERIFY Priority_Array = ACTIVE, ARRAY_INDEX = 6
4. WAIT (approximately 90% of Minimum_On_Time from step 1)
5. VERIFY Priority_Array = ACTIVE, ARRAY_INDEX = 6

Note that in this example, the timing may be such that it cannot be attained while manual testing.

© 2026 by BACnet International. All rights reserved.

2.2.9 WRITE

The WRITE statement is used in a test definition to modify a property value in the IUT. The tester shall generate a WriteProperty-Request and verify that a BACnet-SimpleAck is returned by the IUT by checking the packet log. In some tests the object identifier is omitted from the description because it is clear from the context what object should be written to.

2.2.9.1 Execution Steps

1. Use a test tool to generate the WriteProperty-Request described by the test step.
2. Verify that the IUT returns a BACnet-SimpleAck-Request. If the IUT fails to return a BACnet-SimpleAck-Request, then the IUT shall fail the test. The tester shall report the failure in the Results section for the test along with the test step number.

2.2.9.2 Example

In the following example, the WRITE statement is used to set the Present_Value property of a binary object. The specific object being modified is implied by a previous test step, or by the test parameters documented in the IUT's Test Plan.

11. WRITE Present_Value = ACTIVE, PRIORITY = 7

2.2.10 VERIFY

The VERIFY statement is used in a test definition to check that the value of a property in the IUT is as required by the test.

When verifying property values the precision of the property should be taken into account. This information is usually included in the EPICS.

2.2.10.1 Execution Steps

1. Generate a ReadProperty-Request for the property in question.
2. Verify that the IUT returns a ComplexAck-Request and that the value contained in the request matches the value specified in the test step. If the IUT fails to return a valid ComplexAck-Request, or the value of return in the ComplexAck differs from that in the test step, the IUT shall fail the test. The tester shall report the failure in the Results section for the test along with the test step number.

2.2.10.2 Example

If the following example, the tester would generate a ReadProperty-Request for the Present-Value of the implied object. The tester would then scan the packet log for the response from the IUT and verify that the value returned is 'ACTIVE'.

11. VERIFY Present_Value = ACTIVE

2.2.11 BEFORE

The BEFORE statement is used in a test definition to check that the IUT exhibits a specific behaviour before a specific amount of time passes.

The majority of the occurrences of this statement are followed by a RECEIVE statement. In these cases, the tester shall verify that the packet required by the RECEIVE statement is received within the specified time frame. If the packet is not received within the time frame, then the test shall fail.

Some of the tests use the BEFORE statement to ensure that a specific packet is *not* received within the specified time frame. For test steps of this form, the tester shall verify that the specific packets are not sent by the IUT.

There are also test steps where the BEFORE statement is used to ensure that the tester performs an action within a specified time frame. The tester shall ensure that the action is performed within the time frame. If the tester is unable to perform the action within the specified time frame then the test shall be skipped and this should be noted in the Results section of the IUT's Test Plan.

© 2026 by BACnet International. All rights reserved.

2.2.11.1 Execution Steps

For BEFORE statements followed by a RECEIVE statement:

1. Perform the RECEIVE statement as normal, but wait no longer than the time period described by the BEFORE statement. If the packet is not received within the specified time frame the IUT shall fail the test. The tester shall report the failure in the Results section for the test along with the test step number.

For BEFORE test steps meant to verify that the IUT does not generate a specific packet:

1. Wait the specified time period and the verify in the packet log that the specified packet was not generated by the IUT.
2. If the packet is in the packet log, then the IUT shall fail the test. The tester shall report the failure in the Results section for the test along with the test step number.

For BEFORE test steps that are used to restrict the timing of a tester's actions, the tester shall ensure that the test step is executed within the required time frame.

2.2.11.2 Examples

An example of the BEFORE statement followed by a RECEIVE statement.

```
2.  BEFORE (60 seconds)
3.          RECEIVE PORT A,
                    DESTINATION = LOCAL BROADCAST,
                    SOURCE = IUT,
                    I-Am-Router-To-Network,
                    Network Numbers =       2
```

An example of the BEFORE statement used to verify that the IUT does not generate a specific packet.

```
6.  BEFORE (Acknowledgement Fail Time) {
7.          IF (ReadProperty-ACK received) THEN
8.                  ERROR "Incorrect MS/TP Frame Data CRC undetected."
    }
```

An example of the BEFORE statement used to restrict the timing of the next action of the tester.

```
11. BEFORE Minimum_On_Time
12.         WRITE Present_Value = INACTIVE, PRIORITY = (any value numerically
                    lower than 6 (higher priority))
```

2.2.12 WHILE

The WHILE statement is used to repeatedly perform a test step while a specific condition exists. The tester shall continue to perform the contained statements until the WHILE condition no longer exists.

2.2.12.1 Execution Steps

1. Evaluate the WHILE condition. In some cases, the tester will be required to read property values from the IUT in order to evaluate the WHILE condition. To do so, the tester should initiate a BACnet request using a test tool to determine the property value. If, while evaluating the WHILE condition, the IUT fails to correctly respond to a BACnet request, then the IUT shall fail the test. The tester shall report the failure in the Results section for the test along with the test step number.
2. If the condition evaluates to TRUE, execute the test steps contained within the WHILE statement's body. Repeat from step 1.
3. If the condition evaluates to FALSE, skip the test steps contained within the WHILE statement's body and move on to the next test step.

© 2026 by BACnet International. All rights reserved.

2.2.12.2 Example

In the following example, the WHILE condition is used to wait until the IUT collects enough Trend Log samples to fill its buffer. In this case, the tester is watching the Trend Log object for a specific condition.

6. WHILE ((Total_Record_Count – (value X returned in step 4)) modulo 2^{32} < Buffer_Size) DO {}

2.3 Verifying Received Packets

In order to verify packets a protocol analyzer is used to capture the BACnet traffic on the network. The protocol analyzer contains a log of all packets sent to and from the IUT. A thorough understanding of decoded BACnet messages is required in order to verify packet logs. It is recommended that a 3rd-party protocol analyzer be used to manually verify packets because a decoder developed in-house may suffer from the same potential encoding or decoding errors as the product being tested.

3 Manual Tests

This section describes the test steps for specific tests that cannot otherwise be determined using the general manual test method.

3.1 EPICS Consistency Tests

Review the EPICS for the IUT to verify that all the requirements specified in the EPICS Consistency Tests are met.

3.2 135.1 - 7.1.1 Read Support Test Procedure

Execute a read property request for each property in the test database.

> Using a trusted tool that can send read-property requests to each property in the test database verify that each property read returns a correctly encoded complex-acknowledgement. For each supported property verify the IUT does not error, abort or reject the request.

3.3 135.1 - 7.2.2 Write Support Test Procedure

For each writable property, multiple values will be selected by the tester as defined in the 7.2.1 subclauses to verify the range of supported values. For each property in the test database write property requests are sent to verify the device correctly sends a BACnet-SimpleAck to the request. A subsequent read is made to verify the requested value (as specified in the range requirements) was successfully written and returned by the device.

© 2026 by BACnet International. All rights reserved.

Version	Date	Author	Change
4.0.0	13-Sep-2006	Carl Neilson	• Changed revision numbering
4.0.1	28-Mar-2008	Lori Tribble	• Updated content for review by BTL-WG.
4.0.2	22-May-2008	Lori Tribble	• Added comment about 135.1-5 section for review by BTL-WG.
5.0.1	3-Nov-2008	Lori Tribble	• Updated revision from 4.0.4 to 5.0.1
5.0.2	24-Feb-2009	Lori Tribble	• Updated revision from 4.0.4 to 5.0.1
9.0.1	13-Oct-2011	Duffy O'Craven	• Updated from 5.0.2 to 9.0.1 without change.
9.0.final	01-Dec-2011	Duffy O'Craven	• Updated BACnet International fax number, then to 9.0.final
12.0.final	07-Aug-2012	Duffy O'Craven	• Updated to 12.0.final without change.
14.0.final	17-Nov-2014	Duffy O'Craven	• Updated to 14.0.final without change.
15.0.01	25-Sep-2017	Lori Tribble	• Removed old section 3.1 and wrote new 3.1 opening paragraph.
15.0.final	25-Sep-2017	Lori Tribble	• Accepted all changes and renamed to final
15.0.final	11-Oct-2017	Lori Tribble	• Applied additional errata from voting members.
15.1.final	15-Feb-2018	Lori Tribble	• Updated version in preparation for release of 15.1
15.2.final	31-Oct-2018	Lori Tribble	• Updated version in preparation for release of 15.2
16.0.1	19-Aug-2019	Lori Tribble	• Converted to docx. General cleanup.
16.0.final	25-Sep-2019	Lori Tribble	• Renamed to Final
16.0.final.v2	11-Nov-2019	Emily Hayes	• Renamed to final.v2
16.1	10-Dec-2019	Lori Tribble	• Renamed to 16.1
18.0	18-Oct-2020	Emily Hayes	• Renamed to 18.0
18.1	26-Jan-2021	Emily Hayes	• Renamed to 18.1
20.0	17-Jan-2022	Emily Hayes	• Renamed 20.0
23.0	08-Dec-2022	Emily Hayes	• Renamed 23.0
26.0			• Updated Tools Required clause and removed references to VTS. • Renamed to 26.0
27.0	01-Nov-2026	Emily Hayes	• Renamed 27.0

© 2026 by BACnet International. All rights reserved.