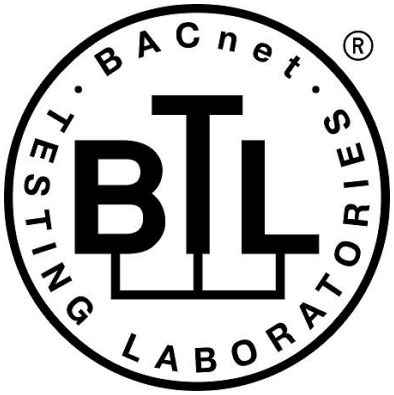


BACnet® TESTING LABORATORIES

Revision 27.0 final
SPECIFIED TESTS

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 8
5 EPICS CONSISTENCY TESTS 8
7. OBJECT SUPPORT TESTS 10
7.2 Write Support for Properties in the Test Database 10
7.2.9 DateTime Non-Pattern Properties Test 10
7.3 Object Functionality Tests 10
7.3.1 Property Tests 10
7.3.1.1 Out_Of_Service, Status_Flags, and Reliability Tests 10
7.3.1.1.1 Out_Of_Service, Status_Flags, and Reliability Test 10
7.3.1.8 Change of State Tests 11
7.3.1.10 Event_Enable Tests 12
7.3.1.10.2 Event_Enable Tests for TO_NORMAL only Algorithms 12
7.3.1.11 Acked_Transitions Tests 14
7.3.1.11.1 Acked_Transitions Test 14
7.3.1.11.3 Acked_Transitions Test for Normal-To-Normal Transitions 17
7.3.1.11.4 Acked_Transitions Test for Normal to Fault Transitions 19
7.3.1.12 Notify_Type Test 21
7.3.1.17 Event_Message_Texts Tests 22
7.3.1.18 Event_Message_Texts_Config Test 24
7.3.1.19 Event_Algorithm_Inhibit Tests 25
7.3.1.19.1 Event_Algorithm_Inhibit Test 25
7.3.1.21 Reliability_Evaluation_Inhibit Tests 27
7.3.1.21.1 Reliability_Evaluation_Inhibit Test 27
7.3.1.21.3 Reliability_Evaluation_Inhibit Object Test 28
7.3.1.22 Event_Detection_Enable Tests 29
7.3.1.22.1 Event_Detection_Enable Inhibits Event Generation 29
7.3.1.27 Blink Warn Tests 30
7.3.1.27.1 Blink-Warn WARN Command Test 30
7.3.1.27.2 Blink-Warn WARN_OFF Command Test 30
7.3.1.27.3 Blink-Warn WARN_RELINQUISH Command Test 31
7.3.1.27.4 Blink-Warn STOP Command Test 32
7.3.1.27.5 Blink-Warn WARN Command Failure Test 33
7.3.1.27.6 Blink-Warn WARN_OFF Command Failure Test 34
7.3.1.27.7 Blink-Warn WARN_RELINQUISH Command Failure Test 35
7.3.1.27.8 Blink-Warn WARN_OFF Command Halted Test 36
7.3.1.27.9 Blink-Warn WARN_RELINQUISH Command Halted Test 38
7.3.1.28 Value Source Mechanism Tests 39
7.3.1.28.1 Writing to the Value_Source Property by a Device Other than the Device that Commanded the Object 39
7.3.1.28.2 Non-commandable Value_Source Property Test 40
7.3.1.28.3 Value_Source Property None Test 40
7.3.1.28.4 Commandable Value Source Test 41
7.3.1.28.5 Life Safety Value_Source Property Test 42
7.3.1.28.X1 Value Source Initiated Locally Test 42
7.3.1.X73 Number_Of_States Property Tests 43
7.3.1.X73.1 Writable Number_Of_States Test 43
7.3.2 Object Specific Tests 43
7.3.2.4 Averaging Object Tests 43
7.3.2.4.1 Reinitializing the Samples 43
7.3.2.6 Binary Output Object Tests 44
7.3.2.9 Command Object Tests 45
7.3.2.9.1 All Writes Successful with Post Delay Test 45
7.3.2.9.2 Quit on Failure Test 45
7.3.2.9.4 Empty Action List Test 46

© 2026 by BACnet International. All rights reserved.

7.3.2.9.5 Action 0 Test 47
7.3.2.9.7 Write While In_Process is TRUE Test 47
7.3.2.10 Device Object Tests 48
7.3.2.10.1 Active_COV_Subscriptions SubscribeCOV Test 48
7.3.2.10.2 Active_COV_Subscriptions SubscribeCOVProperty Test 50
7.3.2.21 Notification Class Object 52
7.3.2.21.3 Recipient_List Tests 52
7.3.2.21.3.1 ~~ValidDays~~*Valid Days* Test 52
7.3.2.21.3.2 ~~FromTime and ToTime Test~~*From Time and To Time Test* 54
7.3.2.21.3.5 Recipient_List Property Supports Device Identifier Recipients Test 55
7.3.2.21.3.6 Recipient_List Property Supports Network Address Recipients 56
7.3.2.21.3.X1 Transitions Test Normal-to-Offnormal 56
7.3.2.21.3.X2 Transitions Test Normal-to-Fault 58
7.3.2.21.3.X3 Transitions Test Normal-to-Normal 60
7.3.2.23 Schedule Object Tests 61
7.3.2.24 Logging Object Tests 61
7.3.2.24.8 Record_Count Test 61
7.3.2.25 Event Log Tests 62
7.3.2.25.1 Internal Logging of Notifications 62
7.3.2.25.X1 Logging Events Never Transmitted as an APDU 63
7.3.2.30 Notification Forwarder Object Tests 64
7.3.2.30.8 Subscribed_Recipients Property Test 64
7.3.2.30.8.1 Time Count Down Test 64
7.3.2.30.13 Persistence Tests 65
7.3.2.30.13.1 Recipient_List Persistence Test 65
7.3.2.30.13.2 Subscribed_Recipients Persistence Test 65
7.3.2.34 Channel Object Tests 66
7.3.2.34.4 Write_Status Test 66
7.3.2.39 Lighting Output Object Tests 67
7.3.2.39.4 Lighting Command Operation FADE_TO Test 67
7.3.2.39.5 Lighting Command Operation RAMP_TO Test 68
7.3.2.39.11 Transition Test 70
7.3.2.40 Access Door Object Tests 70
7.3.2.40.6 Door_Unlock_Delay_Time Test 71
7.3.2.40.X1 Present_Value and Lock_Status Test 72
7.3.2.41 Access Point Object Tests 73
7.3.2.41.3 Denied Access Test 73
7.3.2.41.4 Authorization Mode Test 74
7.3.2.41.7 Lockout State Test 76
7.3.2.41.10 Denied Access Disabled Credential Test 77
7.3.2.41.X1 Resizable Authentication Policy Properties Test 78
7.3.2.42 Access Zone Object Tests 79
7.3.2.42.4 Passback Mode Tests 79
7.3.2.42.4.X1 PASSBACK_OFF Mode Test 80
7.3.2.42.4.X2 SOFT_PASSBACK Mode Test 81
7.3.2.42.4.X3 HARD_PASSBACK Mode Test 82
7.3.2.42.4.X4 Passback Exemption Test 82
7.3.2.46 Network Port Object Tests 83
7.3.2.46.1 Network Port Configuration Tests 83
7.3.2.46.2 Network-Number-Is Updates Network_Number_Quality Test 83
7.3.2.46.3 Network Port Command Tests 84
7.3.2.46.3.2 DISCARD_CHANGES Command Tests 84
7.3.2.46.3.2.2 DISCARD_CHANGES Command Failure Test 84
7.3.2.46.3.3 RENEW_FD_REGISTRATION Command Tests 85
7.3.2.46.3.3.2 RENEW_FD_REGISTRATION Command Failure Test 85
7.3.2.46.3.3.X1 RENEW_FD_REGISTRATION Command Disable Test 86

© 2026 by BACnet International. All rights reserved.

7.3.2.46.3.4 RESTART_SUBORDINATE_DISCOVERY Command Tests 86
7.3.2.46.3.4.1 RESTART_~~SLAVE~~*SUBORDINATE*_DISCOVERY Command Test 86
7.3.2.46.3.4.2 RESTART_~~SLAVE~~*SUBORDINATE*_DISCOVERY Command Failure Test 87
7.3.2.46.3.4.X1 RESTART_SUBORDINATE_DISCOVERY Command Disable Test 87
7.3.2.46.3.5 RENEW_DHCP Command Tests 88
7.3.2.46.3.5.2 RENEW_DHCP Command Failure Test 88
7.3.2.46.3.5.X1 RENEW_DHCP Command Disable Test 89
7.3.2.46.3.6 RESTART_AUTONEGOTIATION Command Tests 89
7.3.2.46.3.6.2 RESTART_AUTONEGOTIATION Command Failure Test 89
7.3.2.46.3.6.X1 RESTART_AUTONEGOTIATION Command Disable Test 90
7.3.2.46.3.7 DISCONNECT Command Tests 90
7.3.2.46.3.7.2 DISCONNECT Command Failure Test 90
7.3.2.46.3.10 Certificate Configuration Tests 91
7.3.2.46.3.10.2 GENERATE_CSR_FILE Command Failure Test 91
7.3.2.46.3.10.4 VALIDATE_CHANGES Command Failure Test 91
7.3.2.46.3.X1 RESTART_DEVICE_DISCOVERY Command Tests 92
7.3.2.46.3.X1.1 RESTART_DEVICE_DISCOVERY Command Test for Non-MSTP Networks 92
7.3.2.46.3.X1.2 RESTART_DEVICE_DISCOVERY Command Test for MSTP Networks 93
7.3.2.46.3.X1.3 RESTART_DEVICE_DISCOVERY Command Failure Test 94
7.3.2.46.3.X1.X1 RESTART_DEVICE_DISCOVERY Command Disable Test 95
7.3.2.46.4 Hierarchical Network Port Tests 95
7.3.2.46.7 DHCP Tests 95
7.3.2.46.7.1 Basic IPv4 DHCP Test 95
7.3.2.46.7.2 Basic IPv6 DHCP Test 96
7.3.2.46.X Valid Hierarchy Tests 97
7.3.2.46.X.1 Valid Hierarchy Chain - MSTP Test 97
7.3.2.46.X.2 Valid Hierarchy Chain - IPvX Test 97
7.3.2.46.X.3 Valid Hierarchy Chain - Application and Physical Test 98
7.3.2.46.X.4 Valid Hierarchy Chain - Secure Connect Test 99
7.3.2.46.X.5 Valid Hierarchy Chain - Proprietary Test 100
7.3.2.46.X.6 Valid Hierarchy Chain - Virtual Test 101
7.3.2.47 Timer Object 102
7.3.2.47.1 Positive Tests 102
7.3.2.47.1.2 Timer Running then Expired Test 102
7.3.2.47.1.12 Forcing Timer Expiration by Writing IDLE 103
7.3.2.50 Staging Object Tests 104
7.3.2.50.11 Out_Of_Service, Status_Flags, and Reliability for Staging Object 104
8. APPLICATION SERVICE INITIATION TESTS 105
8.2 ConfirmedCOVNotification Service Initiation Tests 105
8.2.1 Change of Value Notification for Changes to Present_Value in Objects with a COV_Increment 105
8.2.19 Change of Value Notification from Other Standard Object Types (ConfirmedCOVNotification) 106
8.3 UnconfirmedCOVNotification Service Initiation Tests 107
8.3.10 Device Restart Notifications 107
8.3.21 Change of Value Notification from Other Standard Object Types (UnconfirmedCOVNotification) 108
8.4 ConfirmedEventNotification Service Initiation Tests 109
8.4.2 CHANGE_OF_STATE Tests (ConfirmedEventNotification) 109
8.4.8 CHANGE_OF_LIFE_SAFETY Tests (ConfirmedEventNotification) 111
8.4.8.7 Mode Transition Tests when Event State is Maintained 111
8.4.14 UNSIGNED_RANGE Test (ConfirmedEventNotification Test) 114
8.7 GetEnrollmentSummary Service Initiation Tests 117
8.21 ReadRange Service Initiation Tests 117

© 2026 by BACnet International. All rights reserved.

8.21.9 Presents Log Records 117
8.44 Who-Am-I Service Initiation Tests 117
8.44.1 Responds to Who-Is With Who-Am-I While in the Unconfigured State 117
8.45 You-Are Service Initiation Tests 118
8.45.1 Configures Other Device's MAC Address 118
8.45.2 Configures Other Device's Object Instance Number and MAC Address 118
8.45.3 Can Unconfigure Devices 119
9. APPLICATION SERVICE EXECUTION TESTS 119
9.2 ConfirmedCOVNotification Service Execution Tests 119
9.2.2 Negative ConfirmedCOVNotification Execution Tests 119
9.2.2.1 Change of Value Notification Arrives after Subscription has Expired 119
9.10 SubscribeCOV Service Execution Tests 120
9.10.1 Positive SubscribeCOV Service Execution Tests 120
9.10.1.11 Ensuring 5 Concurrent COV Subscribers 120
9.13 AtomicWriteFile Service Execution Tests 122
9.13.1 Positive AtomicWriteFile Service Execution Tests 122
9.13.1.2 Writing to Stream-Based Files 122
9.13.1.2.1 Writing an Entire Stream-Based File 122
9.14 AddListElement Service Execution Tests 122
9.14.2 Negative AddListElement Service Execution Tests 122
9.14.2.2 Adding a List Element With an Invalid Datatype 122
9.14.2.3 An AddListElement Failure Part Way Through a List 123
9.16 CreateObject Service Execution Tests 124
9.16.2 Negative CreateObject Service Execution Tests 124
9.16.2.3 Attempting to Create an Object with an Object Identifier That is Not Creatable by Specifying the Object Identifier 124
9.16.2.4 Attempting to Create an Object with an Object Type Specifier and an Error in the Initial Values 124
9.16.2.5 Attempting to Create an Object with an Object Identifier and an Error in the Initial Values 125
9.16.2.8 Attempting to Create a non-Supported Object Type (by Object Identifier) 127
9.18 ReadProperty Service Execution Tests 127
9.18.1 Positive ReadProperty Service Execution Tests 127
9.18.1.6 Respects max-segments-accepted bit pattern 127
9.18.1.7 Reading Array Properties at Different Array Indexes 128
9.18.2 Negative ReadProperty Service Execution Tests 128
9.18.2.2 Reading Array Properties With An Array Index That Is Out Of Range 128
9.20 ReadPropertyMultiple Service Execution Tests 129
9.20.1 Positive ReadPropertyMultiple Service Execution Tests 129
9.20.1.7 Reading ALL Properties 129
9.20.1.16 ReadPropertyMultiple Array Properties 129
9.20.2 Negative ReadPropertyMultiple Service Execution Tests 130
9.20.2.3 Reading a Single Non-Array Property with an Array Index 130
9.22 WriteProperty Service Execution Tests 131
9.22.1 Positive WriteProperty Service Execution Tests 131
9.22.1.6 Writing NULL to Non-commandable Properties 131
9.22.2 Negative WriteProperty Service Execution Tests 132
9.22.2.1 Writing Non-Array Properties with an Array Index 132
9.22.2.2 Writing Array Properties with an Array Index that is *Greater Than The Length Of The Array*~~Out of Range~~ 133
9.22.2.3 Writing with a Property Value Having the Wrong Datatype 133
9.22.2.X1 Resizing a Writable Array to a Size That Is Not Supported (WriteProperty) 134
9.23 WritePropertyMultiple Service Execution Tests 134
9.23.1 Positive WritePropertyMultiple Service Execution Tests 134
9.23.1.4 Writing Multiple Properties to Multiple Objects 134
9.23.2 Negative WritePropertyMultiple Service Execution Tests 135

© 2026 by BACnet International. All rights reserved.

9.23.2.4 Writing Non-Array Properties with an Array Index 135
9.23.2.5 Writing Array Properties with an Array Index that is *Greater Than The Length Of The Array*~~Out of Range~~ 136
9.23.2.6 Writing with a Property Value Having the Wrong Datatype 136
9.23.2.12 WritePropertyMultiple Reject Test 137
9.23.2.13 Resizing a Writable Fixed Size Array Property Using WritePropertyMultiple Service 138
9.23.2.14 Writing First Element of 'List of Write Access Specifications' with Object Access Error 139
9.23.2.15 Writing First Element of 'List of Write Access Specifications' with a Write Access Error 140
9.23.2.16 WritePropertyMultiple Reject Test for First Element of 'List of Write Access Specifications' 140
9.23.2.17 Writing First Element of 'List of Write Access Specifications' with a Property Access Error 141
9.23.2.19 Date Non-Pattern Properties Test using WritePropertyMultiple Service 142
9.23.2.20 Time Non-Pattern Properties Test using WritePropertyMultiple Service 142
9.23.2.21 DateTime Non-Pattern Properties Test using WritePropertyMultiple Service 143
9.23.2.22 BACnetDateRange Non-Pattern Properties Test using WritePropertyMultiple Service 144
9.23.2.X1 Resizing a Writable Array to a Size That Is Not Supported (WritePropertyMultiple) 145
9.24 DeviceCommunicationControl Service Execution Test 146
9.24.1 Positive DeviceCommunicationControl Service Execution Tests 146
9.24.1.X1 DeviceCommunicationControl Enable-Disable Test 146
9.24.1.X2 DeviceCommunicationControl Restored with Time Expired 147
9.24.1.X3 DeviceCommunicationControl Restored by DeviceCommunicationControl 148
9.24.1.X4 DeviceCommunicationControl Restored by ReinitializeDevice 148
9.24.1.X5 DeviceCommunicationControl Password Not Required 148
9.24.2 Negative DeviceCommunicationControl Service Execution Tests 149
9.33 Who-Is Service Execution Tests 149
9.33.2 Execution of Who-Is Service Requests Originating from a Remote Network 149
9.33.2.1 General Inquiry, Global Broadcast from a Remote Network 149
9.33.2.2 General Inquiry, Remote Broadcast 150
9.33.2.3 General Inquiry, Directed to a Remote Device 150
9.39 General Testing Service Execution 151
9.39.2 Unsupported Unconfirmed Services Test 151
9.42 SubscribeCOVPropertyMultiple Service Execution Tests 151
9.42.2 Negative SubscribeCOVPropertyMultiple Service Execution Tests 151
9.42.2.6 Array Index Provided Is *Greater Than The Current Length Of The Array*~~Out Of Range~~ 151
9.43 Who-Am-I Service Execution Tests 152
9.43.1 Uses Who-Is to Configure Devices Supporting the Who-Am-I Service 152
9.43.2 Configures Other Device's Object Instance Number 153
9.44 You-Are Service Execution Tests 154
9.44.1 Positive Tests 154
9.44.1.1 Configurable Device Object Instance Number 154
9.44.1.2 Configurable MAC Address 154
9.44.1.3 Configurable Device Object Instance Number and MAC Address 155
9.44.1.4 Device Object Instance Number is Configurable Even When MAC Address is Also Sent 156
9.44.1.5 MAC Address is Reconfigurable 157
9.44.1.6 Retains Configuration Through Restarts 157
9.44.1.7 Unconfigurable by You-Are 159
9.44.2 Negative Tests 159
9.44.2.2 Only Accepts Configuration When Received Parameters Match 159
10. NETWORK LAYER PROTOCOL TESTS 161
10.2 Router Functionality Tests 161
10.2.2 Processing Network Layer Messages 161
10.2.2.4.3 Receiving Messages for a Busy Router 161

© 2026 by BACnet International. All rights reserved.

10.2.8 Routers Execute What-Is-Network-Number 161
10.7 Route Binding Tests....162
10.7.3 Router Binding via Who-Is-Router-To-Network 162
10.8 Virtual Routing Functionality Tests....164
10.8.4 Routing of Broadcast APDUs to Virtual Devices 164
10.8.4.1 Broadcasts that Should Be Ignored....164
10.8.4.2 Route Global Broadcast from a Local Device to Virtual Devices 166
10.8.4.3 Route Global Broadcast from a Remote Device to Virtual Devices....168
10.8.4.4 Route Remote Broadcast from a Local Device to Virtual Devices....171
10.8.4.5 Route Remote Broadcast from a Remote Device to Virtual Devices 172
10.8.4.6 Route Global Broadcast Message from a Virtual Device 173
10.8.4.8 Route Remote Broadcast Message from a Virtual Device to a Remote Network 173
12. DATA LINK LAYER PROTOCOLS TESTS 175
12.1 MS/TP State Machine Tests....175
12.1.3 MS/TP Data Link Layer Tests (Alternate) 175
12.1.3.10 Max_Manager Test....175
12.1.3.11 Max_Info_Frames Test....175
12.1.3.X1 Max_Manager Test for Network Port Object 175
12.1.3.X2 Max_Info_Frames Test for Network Port Objects....176
12.3 BACnet/IP Functionality Tests 177
12.3.2 BBMD B/IP Device with a Server Application....177
12.3.2.2 Execute Original-Broadcast-NPDU....177
12.3.2.2.2 Execute Original-Broadcast-NPDU (Two-hop Distribution) 177
12.3.6 Foreign Device Management....178
12.3.6.3 Foreign Device Table Timer Operations 178
12.3.6.3.1 Non-Zero-Duration Foreign Device Table Timer Operations 178
12.3.8 Foreign Device Tests....179
12.3.8.5 Transmits a Broadcast at Startup preceded by Register-Foreign-Device 179
12.3.11 BBMD Configuration Tests – B side 180
12.3.11.4 *IPv4* - Broadcast Distribution Table Configuration via ~~Hostname~~*Host Name* Entries....180
12.4 BACnet/IPv6 Functionality Tests....181
12.4.4 BBMD Tests....181
12.4.4.1 Positive Tests....182
12.4.4.1.1 Original-Broadcast-NPDU....182
12.4.4.1.4 Forwarded-Address-Resolution 183
12.4.4.3 Broadcast Distribution Table Operations....184
12.4.4.3.X1 IPv6 - Broadcast Distribution Table Configuration via Host Name Entries 184
12.4.5 Foreign Device Management Tests 185
12.4.5.1 Execute Register-Foreign-Device....185
12.5 Secure Connect Functionality Tests....186
12.5.1 Basic Node Tests....186
12.5.1.1 Basic Node Positive Tests 186
12.5.1.1.16 Heartbeat-Request Initiation Test 186
12.5.1.2 Basic Node Negative Tests....187
12.5.1.2.10 Node Heartbeat-Request Initialization Failure Test....187
12.5.2 Hub Tests....189
12.5.2.1 Hub Positive Tests....189
12.5.2.1.9 Duplicate Connection Test....189
12.5.2.1.11 SC_Hub_Function_Enable Property Test....191
12.5.2.2 Hub Negative Tests 193
12.5.2.2.5 Malformed BVLC Test for Nodes operating as a Hub 193
12.5.2.2.6 Hub Heartbeat-Request Initialization Failure Test 196
12.5.3 Direct Connect Tests 197
12.5.3.3 Initiating Direct Connect Tests....197
12.5.3.3.2 Initiating Direct Connect Negative Tests....197
12.5.3.3.2.3 Rejection of Invalid Certificate Outgoing Connection Test....197

© 2026 by BACnet International. All rights reserved.

13 SPECIAL FUNCTIONALITY TESTS 197
13.2 Time Manager 197
13.5 Subordinate Proxy Tests 197
13.5.1 Manual Subordinate Binding Test 197
13.5.2 Automatic Subordinate Discovery Test 199
13.10 Workstation Scheduling Tests 200
13.10.4 Modify an Exception_Schedule 201
13.10.4.7 Modify an Exception_Schedule by Adding a BACnetSpecialEvent with Period of Choice calendarEntry of choice Date 201
13.10.4.8 Modify an Exception_Schedule by Adding a BACnetSpecialEvent with Period of Choice calendarEntry of Choice DateRange 202
13.10.4.9 Modify an Exception_Schedule by Adding a BACnetSpecialEvent with Period of Choice calendarEntry of Choice WeekNDay 202
13.10.4.10 Modify an Exception_Schedule by Adding a BACnetSpecialEvent with Period of Choice calendarReference 203
13.10.X Modify a Timer Object 203
13.10.X.1 Read and Present a State_Change_Values Property 203
13.10.X.2 Modify a State_Change_Values Property 203
13.X1 Time Series Data Tests 204
13.X1.1 Time Series Data Export Tests 204
13.X1.1.1 Validate Time Series Data Export Test 204
13.X2 Device Proxying Functionality Tests 204
13.X2.1 Device Proxying Test 204
13.X2.2 Device Added When Detected via I-Am 206
13.X2.3 Device Discovery Without I-Am 207
13.X2.4 Update of Last I-Am Time 207
13.X2.5 Revalidation of Proxied Devices After Restart 208
13.X2.6 Automatic Removal of Offline Devices from Device_Address_Proxy_Table 209
13.X2.7 Processing Who-Is Messages from Proxied Network 210
13.X2.8 Sending Who-Is Messages Through the Proxied Network 213
13.X2.9 Entry for Self in Device_Address_Proxy_Table 214
13.X2.10 Proxy Table Cleared when Proxy Enable Set to False 215
13.X2.11 Proxying Multiple Devices Test 216
13.X2.12 I-Am from Non-Proxied Network Not Added to Table 217
13.X2.13 Device I-Am Forwarding Test 218

© 2026 by BACnet International. All rights reserved.

1. PURPOSE

This document contains tests defined by the BTL that are not included in ANSI/ASHRAE Standard 135.1-2025 or are modified versions of tests in 135.1. These tests are used by the BTL testing process and are referenced by the BTL Test Plan document.

Most of the tests defined in this document will be submitted to SSPC 135. Those that are submitted will be removed from future versions of this document as they are accepted/rejected by the SSPC 135 and 135.1 is updated.

Some of the tests are interim tests defined by the BTL because the test tools are not adequate for testing the particular functionality. These tests will be removed once the tests in SSPC 135.1 can be implemented by the BTL. Examples of such tests are the MS/TP tests.

5 EPICS CONSISTENCY TESTS

Reason for change: This checking is already completely done in 5-i so 5-m is just duplicate effort without additional value. it can therefore be deleted.

These tests are static tests of the EPICS and do not involve interrogating the IUT. There are no Configuration Requirements or Test Step sections with TCSL in these tests because the tests are static tests of the EPICS and not tests of the IUT itself.

Each implementation shall be tested to ensure consistency among interrelated data elements. These tests shall include:

(a) All object types required by the specified BIBBs shall be indicated as supported in the Standard Object Types Supported section of the EPICS.

(b) A minimum of one instance of each object type required by the specified BIBBs shall be included in the test database.

(c) The Protocol_Object_Types_Supported property of the Device object in the test database shall indicate support for each object type required by the supported BIBBs.

(d) All application services required by the supported BIBBs shall be indicated as supported in the BACnet Standard Application Services Supported section of the EPICS with Initiate and Execute indicated as required by the supported BIBBs.

(e) The Protocol_Services_Supported property of the Device object in the test database shall indicate support for each application service for which the supported BIBBs requires support for execution of the service.

(f) The object types listed in the Standard Object Types Supported section of the EPICS shall have a one-to-one correspondence with object types listed in the Protocol_Object_Types_Supported property of the Device object contained in the test database. An object type is supported if it can be made to exist in the IUT`s database.

(g) For each object type listed in the Standard Object Types Supported* there shall be at least one object of that type in the test database. It is permissible for there to be no instance of the File object type if File objects are dynamically creatable and come into existence only temporarily during Backup and Restore.
* An object type is supported if it can be made to exist in the IUT's database.

(h) There shall be a one-to-one correspondence between the objects listed in the Object_List property of the Device object and the objects included in the test database. The Object_List property and the test database shall both include all proprietary objects. Properties of proprietary objects that are not required by BACnet Clause 23.4.3 need not be included in the test database.

(i) For each object included in the test database, all required properties for that object as defined in Clause 12 of BACnet shall be present. Standard properties which are not defined for the implemented Protocol_Revision shall not be present. In addition, if any of the properties supported for an object require the conditional presence of other properties, their presence shall be verified.

© 2026 by BACnet International. All rights reserved.

(j) For each property that is required to be writable, or conditionally writable, that property shall be marked as writable or conditionally writable, in the EPICS.

(k) The length of the Protocol_Services_Supported bitstring shall have the number of bits defined for BACnetProtocolServicesSupported for the IUT's declared protocol revision.

(l) The length of the Protocol_Object_Types_Supported bitstring shall have the number of bits defined for BACnetObjectTypesSupported for the IUT's declared protocol revision.

(m) ~~For each object included in the test database, any properties that are deprecated or removed shall not appear after the Protocol_Revision in which the property was deprecated or removed.~~ *This item has been removed.*

(n) If the Protocol_Revision property is present in the Device object and its value is greater than or equal to 14, the Property_List property of each object included in the test database shall have one entry for each property present, including non-standard properties with the exception of Object_Type, Object_Identifier, Object_Name, and Property_List.

(o) If the Segmentation_Supported property in the Device object is SEGMENTED_BOTH or SEGMENTED_RECEIVE, then the value of the Max_Segments_Accepted property of the Device object shall be greater than 1.

(p) For each property that is required to be read-only, that property shall not be marked as writable, or conditionality writable, in the EPICS.

(q) For each property for which the standard limits the value range, the value for the property in the EPICS, if provided, shall be within the allowable range as defined in the property definition.

(r) For each property which has a restricted value range defined in the EPICS, the restricted value range shall be within the allowable value range and contain the minimal value range as defined by the standard.

© 2026 by BACnet International. All rights reserved.

7. OBJECT SUPPORT TESTS

7.2 Write Support for Properties in the Test Database

7.2.9 DateTime Non-Pattern Properties Test

Reason For Change: Remove day of week test based on IR.

Purpose: To verify that the property being tested does not accept special date field values.

Test Concept: The property being tested, P1, is written with each of the special datetime field values to ensure that the property does not accept them. A datetime DT1 is selected which is within the range that the IUT will accept for the property. The value, V1, written to the property, is the datetime DT1 with one of its fields replaced with one of the date or time special values. If the property is a complex datatype, the other fields in the value shall be set within the range accepted by the IUT. *It is a local matter whether the device accepts or rejects an invalid day of week field so it is not tested.* ~~This test shall only be applied to devices claiming Protocol_Revision 11 or higher.~~

~~Notes to Tester: If P1 is an array, then an array index shall be provided in the TRANSMIT portion of step 1.~~

Test Steps:

1. REPEAT SV = (year unspecified, month unspecified, day of month unspecified, ~~day of week unspecified,~~
odd months, even months, last day of month, even days, odd days, hour unspecified, minute unspecified,
second unspecified, hundredths unspecified) DO {

~~1~~*2*. TRANSMIT WriteProperty-Request
'Object Identifier' = O1,
'Property Identifier' = P1,
'Property Value' = (DT1 updated with the special value SV)

~~2~~*3*. RECEIVE BACnet-Error-PDU
'Error Class' = PROPERTY,
'Error Code' = VALUE_OUT_OF_RANGE

}

7.3 Object Functionality Tests

7.3.1 Property Tests

7.3.1.1 Out_Of_Service, Status_Flags, and Reliability Tests

7.3.1.1.1 Out_Of_Service, Status_Flags, and Reliability Test

Reason for Change: Modified test to be more automation friendly. Add PR 20 requirements and test that a change to the Present_Value that results in a change to Reliability is also tested.

Purpose: To verify that Present_Value is writable when Out_Of_Service is TRUE and that the interrelationship between the Out_Of_Service, Status_Flags, and Reliability properties.

Test Concept: The value of the Out_Of_Service property is set to TRUE and the Present_Value property is tested to be writable. *If a value of the Present_Value results in a change to the Reliability property, verify the Status_Flags change. If the Reliability property is present and writable, verify a write to the Reliability property changes the Status_Flags.* ~~The value of the Status_Flags property is validated and, if present, the value of the Reliability property is also validated.~~ The value of the Status_Flags property, ~~SF1,~~ and, if present, the Reliability property, ~~R1,~~ are checked to ensure they return to their initial values when the value of the Out_Of_Service property is set to FALSE.

Configuration Requirements: If the selected object is commandable, the values of the entries in the Priority_Array above the selected priority, PTY1, shall be NULL. *The Reliability property shall be NO_FAULT_DETECTED.*

© 2026 by BACnet International. All rights reserved.

Test Steps:

~~1. READ SF1 = Status_Flags~~
1. READ PV1 = Present_Value
2. VERIFY Status_Flags = (?, FALSE, ?, FALSE)
~~3. IF Reliability is present THEN~~
~~4. READ R1 = Reliability~~
~~6~~*3*. IF (Out_Of_Service is writable) THEN
~~6~~*4*. WRITE Out_Of_Service = TRUE
ELSE
~~7~~*5*. MAKE (Out_Of_Service TRUE)
~~8~~*6*. VERIFY Out_Of_Service = TRUE
~~9~~*7*. VERIFY Status_Flags = (?, ~~?~~*FALSE*, ?, TRUE)
~~10~~*8*.REPEAT X = (all values meeting the functional range requirements of 7.2.1) DO {
~~11~~*9*. WRITE Present_Value, PTY1 = X
~~12~~*10*. VERIFY Present_Value = X
11. IF Reliability is present THEN
-- Check if Reliability changed due to a change to the Present_Value
12. IF (Reliability <> NO_FAULT_DETECTED) THEN
13. VERIFY Status_Flags = (?, TRUE, ?, TRUE)
14. WRITE Present_Value, PTY1 = PV1
15. VERIFY Reliability = NO_FAULT_DETECTED
16. VERIFY Status_Flags = (?, FALSE, ?, TRUE)
}
17. WRITE Present_Value, PTY1 = PV1
18. IF (Reliability is present) THEN
19. IF (((Protocol_Revision >= 20) AND (Reliability can take on a value other than NO_FAULT_DETECTED)) OR (Reliability is writable)) THEN
~~14. IF (Reliability is present and writable) THEN~~
~~15~~*20*. REPEAT X = (all values of the Reliability enumeration *supported by* ~~appropriate to~~ the object type except NO_FAULT_DETECTED) DO {
~~16~~*21*. WRITE Reliability = X
~~17~~*22*. VERIFY Reliability = X
~~18~~*23*. VERIFY Status_Flags = (?, TRUE, ?, TRUE)
~~19~~*24*. WRITE Reliability = NO_FAULT_DETECTED
~~20~~*25*. VERIFY Reliability = NO_FAULT_DETECTED
~~21~~*26*. VERIFY Status_Flags = (?, FALSE, ?, TRUE)
}
~~22~~*27*. IF (Out_Of_Service is writable) THEN
~~23~~*28*. WRITE Out_Of_Service = FALSE
ELSE
~~24~~*29*. MAKE (Out_Of_Service FALSE)
~~25~~*30*. VERIFY Out_Of_Service = FALSE
~~26~~*31*. VERIFY Status_Flags = ~~SF1~~*(?, FALSE, ?, FALSE)*
~~27~~*32*. IF Reliability is present THEN
~~28~~*33*. VERIFY Reliability = ~~R1~~*NO_FAULT_DETECTED*

7.3.1.8 Change of State Tests~~

Reason For Change: Added Note to Tester about commandable objects.

Purpose: To verify that the properties of objects that collectively track state changes (changes in Present_Value) function as required.

© 2026 by BACnet International. All rights reserved.

Test Concept: The Present_Value of the object under test *(O1)* is changed. The Change_Of_State_Count property is checked to verify that it has been incremented, and the Change_Of_State_Time property is checked to verify that it has been updated. The Change_Of_State_Count is reset and Time_Of_State_Count_Reset is checked to verify that it has been updated appropriately.

Configuration Requirements: The object being tested shall be configured such that the Present_Value and Change_Of_State_Count properties are writable, or another means of changing these properties shall be provided.

Notes To Tester: If O1 is a commandable object, ensure the Property Array Index field in the write requests is sufficient to cause the Present_Value property to be updated. Consider using a Property Array Index at a priority numerically lower (higher priority) than priority 6 if Minimum_On_Time and Minimum_Off_Time properties are present.

Test Steps:

1. READ PV = Present_Value
2. READ N = Change_of_State_Count
3. IF (PV = ACTIVE) THEN
4. IF (Present_Value is writable) THEN
5. WRITE Present_Value = INACTIVE
6. VERIFY Present_Value = INACTIVE
   ELSE
7. MAKE (Present_Value = INACTIVE)
   ELSE
8. IF (Present_Value is writable) THEN
9. WRITE Present_Value = ACTIVE
10. VERIFY Present_Value = ACTIVE
    ELSE
11. MAKE (Present_Value = ACTIVE)
12. VERIFY (Change_of_State_Count = N+1)
13. VERIFY (Change_Of_State_Time ~= the current local date and time)
14. IF (Change_of_State_Count is writable) THEN
15. WRITE Change_of_State_Count = 0
    ELSE
16. MAKE (Change_of_State_Count = 0)
17. VERIFY Time_Of_State_Count_Reset ~= (the current local date and time)

7.3.1.10 Event_Enable Tests

7.3.1.10.2 Event_Enable Tests for TO_NORMAL only Algorithms

Reason For Change: Add new Notes to Tester to include a note on how to execute the test with algorithms that do not support the pTimeDelayNormal parameter. Move existing Notes to Tester to exist above Test Steps. Renumber to use new number schema and fix typo in Step 7.

Purpose: To verify that notification messages are transmitted only if the bit in Event_Enable corresponding to the event transition has a value of TRUE. This test applies to objects that only support generation of TO_NORMAL transitions.

Test Concept: The IUT is configured such that the Event_Enable property indicates that some event transitions are to trigger an event notification. Each event transition is triggered, and the IUT is monitored to verify that notification messages are transmitted only for those transitions for which the Event_Enable property has a value of TRUE.

© 2026 by BACnet International. All rights reserved.

Configuration Requirements: In the Notification Class object providing recipient information, the value of the Transitions parameter for all recipients shall be (TRUE, TRUE, TRUE). The event-generating object shall be in a NORMAL state at the start of the test. D1 is either the pTimeDelayNormal parameter or, in case of the EXTENDED and proprietary algorithms, a vendor specific delay (may be zero).

Notes to Tester: For algorithms that do not apply the pTimeDelayNormal parameter, assume D1 equals zero when executing the test.

Notes to Tester: The UnconfirmedEventNotification service may be substituted for the ConfirmedEventNotification service, in which case the tester shall skip all of the steps in which a BACnet-SimpleACK-PDU is sent.

Test Steps:

1. VERIFY pCurrentState = NORMAL
2. MAKE (the TO-NORMAL bit of the Event_Enable property equal to TRUE)
3. MAKE (a condition exist that will cause the object to generate a TO_NORMAL transition)
4. WAIT D1
5. BEFORE Notification Fail Time
6. RECEIVE ConfirmedEventNotification-Request,
   'Process Identifier' = (any valid process ID),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (the event-generating object configured for this test),
   'Time Stamp' = (any valid time stamp),
   'Notification Class' = (the class corresponding to the object being tested),
   'Priority' = (the value configured to correspond to a TO-NORMAL transition),
   'Event Type' = (any valid event type),
   'Message Text' = (optional, any valid message text),
   'Notify Type' = EVENT | ALARM,
   'AckRequired' = TRUE | FALSE,
   'From State' = NORMAL,
   'To State' = NORMAL,
   'Event Values' = (values appropriate to the event type)

~~6.~~*7.* ~~TRANSMITBACnet-SimpleACK-PDU~~*TRANSMIT BACnet-SimpleACK-PDU*

~~7.~~*8.* VERIFY pCurrentState = NORMAL

~~8.~~*9.* IF (Event_Enable can be changed such that the TO-NORMAL transition is FALSE) THEN {

10. MAKE (the TO-NORMAL bit of the Event_Enable property equal to FALSE)

11. MAKE (a condition exist that would cause the object to generate a TO_NORMAL transition)

12. WAIT (D1 + Notification Fail Time)

13. CHECK (verify that the IUT did not transmit an event notification message)

14. MAKE (the TO_NORMAL bit of the Event_Enable property equal to TRUE)

}

~~9.~~*15.* IF (the event-generating object can be placed into a fault condition) THEN {

16. IF (Event_Enable can be modified) THEN

17. MAKE (Event_Enable TO-FAULT transition equal TRUE)

18. IF (Event_Enable TO-FAULT transition = TRUE) THEN {

19. MAKE (the event-triggering object change to a fault condition)

20. BEFORE Notification Fail Time

21. RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (any valid process ID),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the event-generating object configured for this test),
'Time Stamp' = (any valid time stamp),
'Notification Class' = (the class corresponding to the object being tested),
'Priority' = (the value configured to correspond to a TO-FAULT transition),
'Event Type' = (IF (Protocol_Revision < 13) THEN
(any valid event type),

© 2026 by BACnet International. All rights reserved.

ELSE
CHANGE_OF_RELIABILITY),
'Message Text' = (optional, any valid message text),
'Notify Type' = EVENT | ALARM,
'AckRequired' = TRUE | FALSE,
'From State' = NORMAL,
'To State' = FAULT,
'Event Values' = (values appropriate to the event type)
22. TRANSMIT BACnet-SimpleACK-PDU
23. VERIFY pCurrentState = FAULT
24. MAKE (the event-generating object change to a normal condition)
25. BEFORE Notification Fail Time
26. RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (any valid process ID),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the event-generating object configured for this test),
'Time Stamp' = (any valid time stamp),
'Notification Class' = (the class corresponding to the object being tested),
'Priority' = (the value configured to correspond to a TO-NORMAL transition),
'Event Type' = (IF (Protocol_Revision < 13) THEN
(any valid event type),
ELSE
CHANGE_OF_RELIABILITY),
'Message Text' = (optional, any valid message text),
'Notify Type' = EVENT | ALARM,
'AckRequired' = TRUE | FALSE,
'From State' = FAULT,
'To State' = NORMAL,
'Event Values' = (values appropriate to the event type)
27. TRANSMIT BACnet-SimpleACK-PDU
28. VERIFY pCurrentState = NORMAL
}
}
~~10.~~*29.* IF (Event_Enable can be modified) THEN
30. MAKE (Event_Enable TO-FAULT transition equal FALSE)
~~11.~~*31.* IF (Event_Enable TO-FAULT transition = FALSE) THEN {
32. MAKE (the event-triggering object change to a fault condition)
33. VERIFY pCurrentState = FAULT
34. CHECK (verify that the IUT did not transmit an event notification message)
35. MAKE (the event-triggering object change to a normal condition)
}

~~Notes to Tester: The UnconfirmedEventNotification service may be substituted for the ConfirmedEventNotification service, in which case the TD shall skip all of the steps in which a BACnet-SimpleACK-PDU is sent.~~

7.3.1.11 Acked_Transitions Tests

7.3.1.11.1 Acked_Transitions Test

Reason For Change: Add new Notes to Tester to include a note on how to execute the test with the EXTENDED algorithm, which does not support the pTimeDelay or pTimeDelayNormal parameters. Add Test Conditionality for skipping test if the IUT only supports normal to normal event generating objects. Fix small typo in existing Notes to Tester.

© 2026 by BACnet International. All rights reserved.

Purpose: To verify that the Acked_Transitions property tracks whether or not an acknowledgment has been received for a previously issued event notification. It also verifies the interrelationship between Status_Flags and Event_State.

Test Concept: The IUT is configured such that the Event_Enable property indicates that all event transitions are to trigger an event notification. The Acked_Transitions property shall have the value (TRUE, TRUE, TRUE) indicating that all previous transitions have been acknowledged. Each event transition is triggered and the Acked_Transitions property is monitored to verify that the appropriate bit is cleared when a notification message is transmitted and reset if an acknowledgment is received.

Test Conditionality: If the IUT only supports normal to normal event generating objects, then this test shall be skipped.

Configuration Requirements: The Event_Enable and Acked_Transitions properties shall be configured with a value of (TRUE, TRUE, TRUE). For analog objects the Limit_Enable property shall be configured with the value (TRUE, TRUE). The referenced event-triggering property shall be set to a value that results in a NORMAL condition. The value of the Transitions parameter for all recipients shall be (TRUE, TRUE, TRUE).

Notes to Tester: In case of the EXTENDED and proprietary algorithms, apply the vendor specific delay, which may equal zero, for the pTimeDelay and pTimeDelayNormal parameters when executing the test.

Notes to Tester: The UnconfirmedEventNotification service may be substituted for the ConfirmedEventNotification service, in which case the TD shall skip sending the BACnet-SimpleACK-PDU messages after receiving the notifications.

Notes to Tester: For life safety objects that latch pMonitoredValue, the LifeSafetyOperation Service will be required to reset pMonitoredValue.

Test Steps:
1. VERIFY pCurrentState = NORMAL
2. VERIFY Acked_Transitions = (TRUE, TRUE, TRUE)
3. IF (Protocol_Revision is present AND Protocol_Revision >= 13) THEN
4. VERIFY Status_Flags = (FALSE, FALSE, ?, ?)
5. IF (pMonitoredValue is writable) THEN
6. WRITE pMonitoredValue = (a value that is OFFNORMAL)
   ELSE
7. MAKE (pMonitoredValue have a value that is OFFNORMAL)
8. WAIT (pTimeDelay)
9. BEFORE Notification Fail Time
10. RECEIVE ConfirmedEventNotification-Request,
    'Process Identifier' = (~~PI1: any valid process ID~~*any valid process ID*),
    'Initiating Device Identifier' = IUT,
    'Event Object Identifier' = (the event-generating object configured for this test),
    'Time Stamp' = (Toffnormal: any valid time stamp),
    'Notification Class' = (the class corresponding to the object being tested),
    'Priority' = (Poffnormal: the value configured to correspond to a TO-OFFNORMAL transition),
    'Event Type' = (any valid event type),
    'Message Text' = (optional, any valid message text),
    'Notify Type' = (the notify type configured for this event),
    'AckRequired' = TRUE,
    'From State' = NORMAL,
    'To State' = OFFNORMAL,
    'Event Values' = (values appropriate to the event type)
11. TRANSMIT BACnet-SimpleACK-PDU
12. VERIFY pCurrentState = OFFNORMAL
13. VERIFY Acked_Transitions = (FALSE, TRUE, TRUE)
14. IF (Protocol_revision is present AND Protocol_Revision >= 13 THEN
15. VERIFY Status_Flags = (TRUE, FALSE, ?, ?)
16. IF (pMonitoredValue is writable) THEN
17. WRITE pMonitoredValue = (a value that is NORMAL)

© 2026 by BACnet International. All rights reserved.

ELSE
18. MAKE (pMonitoredValue have a value that is NORMAL)
19. WAIT (pTimeDelayNormal)
20. BEFORE Notification Fail Time
21. RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (~~PI2: any valid process ID~~*any valid process ID*),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the event-generating object configured for this test),
'Time Stamp' = (Tnormal: any valid time stamp),
'Notification Class' = (the class corresponding to the object being tested),
'Priority' = (Pnormal: the value configured to correspond to a TO-NORMAL transition),
'Event Type' = (any valid event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = (the notify type configured for this event),
'AckRequired' = TRUE,
'From State' = OFFNORMAL,
'To State' = NORMAL,
'Event Values' = (values appropriate to the event type)
22. TRANSMIT BACnet-SimpleACK-PDU
23. VERIFY pCurrentState = NORMAL
24. VERIFY Acked_Transitions = (FALSE, TRUE, FALSE)
25. IF (Protocol_Revision is present AND Protocol_Revision >= 13) THEN
26. VERIFY Status_Flags = (FALSE, FALSE, ?,?)
27. TRANSMIT AcknowledgeAlarm-Request,
'Acknowledging Process Identifier' = (~~PI2~~*any valid process ID*),
'Event Object Identifier' = (the event-generating object configured for this test),
'Event State Acknowledged' = NORMAL,
'Time Stamp' = (Tnormal),
'Acknowledgement Source' = (a character string),
'Time of Acknowledgment' = (the TD's current time)
28. RECEIVE BACnet-SimpleACK-PDU
29. IF (Protocol_Revision is present and Protocol_Revision >= 1) THEN
30. BEFORE Notification Fail Time
31. RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (~~PI2~~*any valid process ID*),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the event-generating object configured for this test),
'Time Stamp' = (the IUT's current time or sequence number),
'Notification Class' = (the class corresponding to the object being tested),
'Priority' = (Pnormal),
'Event Type' = (any valid event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = ACK_NOTIFICATION,
'To State' = NORMAL
ELSE
32. BEFORE Notification Fail Time
33. RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (~~PI2~~*any valid process ID*),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the event-generating object configured for this test),
'Time Stamp' = (the IUT's current time or sequence number),
'Notification Class' = (the class corresponding to the object bind tested),
'Priority' = (Pnormal),
'Event Type' = (any valid event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = ACK_NOTIFICATION

© 2026 by BACnet International. All rights reserved.

34. TRANSMIT BACnet-SimpleACK-PDU
35. VERIFY Acked_Transitions = (FALSE, TRUE, TRUE)
36. TRANSMIT AcknowledgeAlarm-Request,
        'Acknowledging Process Identifier' = (~~PI1~~*any valid process ID*),
        'Event Object Identifier' = (the event-generating object configured for this test),
        'Event State Acknowledged' = OFFNORMAL,
        'Time Stamp' = (Toffnormal),
        'Acknowledgement Source' = (a character string),
        'Time of Acknowledgment' = (the TD's current time)
37. RECEIVE BACnet-SimpleACK-PDU
38. IF (Protocol_Revision is present and Protocol_Revision >= 1) THEN
39.   BEFORE Notification Fail Time
40.   RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' = (~~PI1~~*any valid process ID*),
                'Initiating Device Identifier' = IUT,
                'Event Object Identifier' = (the event-generating object configured for this test),
                'Time Stamp' = (the IUT's current time or sequence number),
                'Notification Class' = (the class corresponding to the object being tested),
                'Priority' = (Poffnormal),
                'Event Type' = (any valid event type),
                'Message Text' = (optional, any valid message text),
                'Notify Type' = ACK_NOTIFICATION,
                'To State' = OFFNORMAL
    ELSE
41.   BEFORE Notification Fail Time
42.   RECEIVE ConfirmedEventNotification-Request,
                'Process Identifier' = (~~PI1~~*any valid process ID*),
                'Initiating Device Identifier' = IUT,
                'Event Object Identifier' = (the event-generating object configured for this test),
                'Time Stamp' = (the IUT's current time or sequence number),
                'Notification Class' = (the class corresponding to the object being tested),
                'Priority' = (Poffnormal),
                'Event Type' = (any valid event type),
                'Message Text' = (optional, any valid message text),
                'Notify Type' = ACK_NOTIFICATION
43. TRANSMIT BACnet-SimpleACK-PDU
44. VERIFY Acked_Transitions = (TRUE, TRUE, TRUE)

7.3.1.11.3 Acked_Transitions Test for Normal-To-Normal Transitions

Reason For Change: Add new Notes to Tester to indicate how to execute the tests with algorithms that do not support the pTimeDelay or pTimeDelayNormal parameters. Fix numbering issue.

Purpose: To verify that the Acked_Transitions property tracks whether an acknowledgment has been received for a previously issued normal event notification. It also verifies the interrelationship between Status_Flags and Event_State.

Test Concept: The IUT is configured such that the Event_Enable property indicates that normal event transitions are to trigger an event notification. The normal event transition is triggered and the Acked_Transitions property is monitored to verify that the appropriate bit is cleared when a notification message is transmitted and reset when an acknowledgment is received.

Configuration Requirements: The Event_Enable and Acked_Transitions properties shall be configured with a value of (?, ?, TRUE). The referenced event-triggering property shall be set to a value that results in a NORMAL condition. The value of the Transitions parameter for all recipients shall be (?, ?, TRUE).

© 2026 by BACnet International. All rights reserved.

Notes to Tester: For algorithms that do not apply the pTimeDelay or pTimeDelayNormal parameters, assume pTimeDelay or pTimeDelayNormal equals zero when executing the test.

Notes to Tester: The UnconfirmedEventNotification service may be substituted for the ConfirmedEventNotification service, in which case the TD shall skip sending the BACnet-SimpleACK-PDU messages after receiving the notification.

Notes to Tester: For life safety objects that latch pMonitoredValue, the LifeSafetyOperation Service will be required to reset pMonitoredValue.

Test Steps:

1. VERIFY pCurrentState = NORMAL
2. VERIFY Acked_Transitions = (?, ?, TRUE)
3. IF (Protocol_Revision is present AND Protocol_Revision >= 13) THEN
4. VERIFY Status_Flags = (FALSE, FALSE, ?, ?)
~~4.~~*5.* IF (pMonitoredValue is writable) THEN
~~5.~~*6.* WRITE pMonitoredValue = (a value that will result in a TO_NORMAL transition)
ELSE
~~6.~~*7.* MAKE (pMonitoredValue a value that will result in a TO_NORMAL transition)
~~7.~~*8.* WAIT (pTimeDelayNormal)
~~8.~~*9.* BEFORE Notification Fail Time
~~9.~~*10.* RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (~~PI2: any valid process ID~~*any valid process ID*),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the event-generating object configured for this test),
'Time Stamp' = (Tnormal: any valid time stamp),
'Notification Class' = (the class corresponding to the object being tested),
'Priority' = (Pnormal: the value configured to correspond to a TO-NORMAL transition),
'Event Type' = (any valid event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = (the notify type configured for this event),
'AckRequired' = TRUE,
'From State' = NORMAL,
'To State' = NORMAL,
'Event Values' = (values appropriate to the event type)
~~10.~~*11.* TRANSMIT BACnet-SimpleACK-PDU
~~11.~~*12.* VERIFY pCurrentState = NORMAL
~~12.~~*13.* VERIFY Acked_Transitions = (?, ?, FALSE)
~~13.~~*14.* IF (Protocol_Revision is present AND Protocol_Revision >= 13) THEN
~~14.~~*15.* VERIFY Status_Flags = (FALSE, FALSE, ?,?)
~~15.~~*16.* TRANSMIT AcknowledgeAlarm-Request,
'Acknowledging Process Identifier' = (PI2),
'Event Object Identifier' = (the event-generating object configured for this test),
'Event State Acknowledged' = NORMAL,
'Time Stamp' = (Tnormal),
'Acknowledgement Source' = (a character string),
'Time of Acknowledgment' = (the TD’s current time)
~~16.~~*17.* RECEIVE BACnet-SimpleACK-PDU
~~17.~~*18.* IF (Protocol_Revision is present AND Protocol_Revision >= 1) THEN
~~18.~~*19.* BEFORE Notification Fail Time
~~19.~~*20.* RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (~~PI2~~*any valid process ID*),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the event-generating object configured for this test),
'Time Stamp' = (the IUT's current time or sequence number),
'Notification Class' = (the class corresponding to the object being tested),

© 2026 by BACnet International. All rights reserved.

'Priority' = (Pnormal),
'Event Type' = (any valid event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = ACK_NOTIFICATION,
'To State' = NORMAL

ELSE

~~20.~~*21.* BEFORE Notification Fail Time

~~21~~*22.* RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (~~PI2~~*any valid process ID*),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the event-generating object configured for this test),
'Time Stamp' = (the IUT's current time or sequence number),
'Notification Class' = (the class corresponding to the object being tested),
'Priority' = (Pnormal),
'Event Type' = (any valid event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = ACK_NOTIFICATION

~~22.~~*23.* TRANSMIT BACnet-SimpleACK-PDU

~~23.~~*24.* VERIFY Acked_Transitions = (?, ?, TRUE)

7.3.1.11.4 Acked_Transitions Test for Normal to Fault Transitions

Reason for Change: New test.

Purpose: To verify that the Acked_Transitions property tracks whether or not an acknowledgment has been received for a previously issued fault event notification. It also verifies the interrelationship between Status_Flags and Event_State.

Test Concept: The IUT is configured such that the Event_Enable property indicates that fault event transitions are to trigger an event notification. The Acked_Transitions property shall have the value (?, TRUE, ?). The fault event transition is triggered and the Acked_Transitions property is monitored to verify that the appropriate bit is cleared when a notification message is transmitted and reset when an acknowledgment is received.

Configuration Requirements: The Event_Enable and Acked_Transitions properties shall be configured with a value of (?, TRUE, ?). The referenced event-triggering property shall be set to a value that results in a NORMAL condition. The value of the Transitions parameter for all recipients shall be (?, TRUE, ?).

Notes to Tester: The UnconfirmedEventNotification service may be substituted for the ConfirmedEventNotification service, in which case the TD shall skip sending the BACnet-SimpleACK-PDU messages after receiving the notifications.

Test Steps:

1. VERIFY pCurrentState = NORMAL
2. VERIFY Acked_Transitions = (?, TRUE, ?)
3. IF (Protocol_Revision is present AND Protocol_Revision >= 13) THEN
4. VERIFY Status_Flags = (FALSE, FALSE, ?, ?)
5. MAKE (a condition exist that will cause the object to generate a fault condition)
6. BEFORE **Notification Fail Time**
7. RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (PI3: any valid process ID),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the event-generating object configured for this test),
'Time Stamp' = (Tfault: any valid time stamp),
'Notification Class' = (the class corresponding to the object being tested),
'Priority' = (Pfault: the value configured to correspond to a TO-FAULT transition),
'Event Type' = IF (Protocol_Revision < 13) THEN

© 2026 by BACnet International. All rights reserved.

(any valid event type),
ELSE
CHANGE_OF_RELIABILITY,
'Message Text' = (optional, any valid message text),
'Notify Type' = (the notify type configured for this event),
'AckRequired' = TRUE,
'From State' = NORMAL,
'To State' = FAULT,
'Event Values' = (values appropriate to the event type)

8. TRANSMIT BACnet-SimpleACK-PDU
9. VERIFY pCurrentState = FAULT
10. VERIFY Acked_Transitions = (?, FALSE, ?)
11. IF (Protocol_revision is present AND Protocol_Revision >= 13 THEN
12. VERIFY Status_Flags = (~~FALSE~~?, TRUE, ?, ?)
13. TRANSMIT AcknowledgeAlarm-Request,
'Acknowledging Process Identifier' = (PI3),
'Event Object Identifier' = (the event-generating object configured for this test),
'Event State Acknowledged' = FAULT,
'Acknowledgement Source' = (a character string),
'Time Stamp' = (Tfault),
'Time of Acknowledgment' = (the TD's current time)
14. RECEIVE BACnet-SimpleACK-PDU
15. IF (Protocol_Revision is present AND Protocol_Revision ≥ 1) THEN
16. BEFORE **Notification Fail Time**
17. RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (PI3),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the event-generating object configured for this test),
'Time Stamp' = (Tfault the IUT's current time or sequence number),
'Notification Class' = (the class corresponding to the object being tested),
'Priority' = (Pfault),
'Event Type' = IF (Protocol_Revision < 13)
(any valid event type),
ELSE
CHANGE_OF_RELIABILITY,
'Message Text' = (optional, any valid message text),
'Notify Type' = ACK_NOTIFICATION,
'To State' = FAULT

ELSE

18. BEFORE **Notification Fail Time**
19. RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (PI3),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the event-generating object configured for this test),
'Time Stamp' = (Tfault the IUT's current time or sequence number),
'Notification Class' = (the class corresponding to the object being tested),
'Priority' = (Pfault),
'Event Type' = (any valid event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = ACK_NOTIFICATION
20. TRANSMIT BACnet-SimpleACK-PDU
21. VERIFY Acked_Transitions = (?, TRUE, ?)

© 2026 by BACnet International. All rights reserved.

7.3.1.12 Notify_Type Test

Reason For Change: Add new Notes to Tester to include a note on how to execute the tests with algorithms that do not support pTimeDelay or pTimeDelayNormal. Add a period to the sentence at end of Configuration Requirement's first paragraph. Renumber test steps to new numbering schema.

Purpose: To verify that the value of the Notify_Type property determines whether an event notification is transmitted as an alarm or as an event.

Configuration Requirements: The IUT shall be configured with two event-generation objects, E1 and E2. Object E1 shall be configured with a Notify_Type of ALARM and E2 shall be configured with a Notify_Type of EVENT. ~~The value of the Transitions parameter for all recipients shall be (TRUE, TRUE, TRUE)~~*The value of the Transitions parameter for all recipients shall be (TRUE, TRUE, TRUE).*

In the test description below X1 and X2 are used to designate the pMonitoredValue algorithm parameter linked to E1 and E2 respectively. X1 and X2 shall be set to a value that results in a NORMAL condition of E1 and E2, respectively.

Notes to Tester: For algorithms that do not apply the pTimeDelay or pTimeDelayNormal parameters, assume pTimeDelay or pTimeDelayNormal equals zero when executing the test.

Notes to Tester: If Notify_Type is writable this test may be performed with one event generating object by changing Notify_Type from ALARM to EVENT in order to cover both cases.

Test Steps:

1. VERIFY (E1), pCurrentState = NORMAL
2. VERIFY (E2), pCurrentState = NORMAL
3. WAIT (pTimeDelay + Notification Fail Time)
4. IF (X1 is writable) THEN
5. WRITE X1 = (a value that will cause a transition in E1)
   ELSE
6. MAKE (X1 a value that will cause a transition in E1)

~~5.~~7. IF (the transition is not a FAULT transition) THEN
8. IF (the transition is a TO-OFFNORMAL transition) THEN
9. WAIT (pTimeDelay)
   ELSE
10. WAIT (pTimeDelayNormal)

~~6.~~11. BEFORE Notification Fail Time
12. RECEIVE ConfirmedEventNotification-Request,
    'Process Identifier' = (any valid process ID),
    'Initiating Device Identifier' = IUT,
    'Event Object Identifier' = (E1),
    'Time Stamp' = (any valid time stamp),
    'Notification Class' = (the class corresponding to the object being tested),
    'Priority' = (any valid value),
    'Event Type' = (any valid event type),
    'Message Text' = (optional, any valid message text),
    'Notify Type' = ALARM,
    'AckRequired' = TRUE | FALSE,
    'From State' = NORMAL,
    'To State' = (any valid value),
    'Event Values' = (values appropriate to the event type)

~~7.~~13. TRANSMIT BACnet-SimpleACK-PDU

~~8.~~14. IF (X2 is writable) THEN
15. WRITE X2 = (a value that will cause a transition in E2)
    ELSE
16. MAKE (X2 a value that will cause a transition in E2)

© 2026 by BACnet International. All rights reserved.

~~9.~~*17.* IF (the transition is not a FAULT transition) THEN
18. IF (the transition is a TO-OFFNORMAL transition) THEN
19. WAIT (pTimeDelay)
ELSE
20. WAIT (pTimeDelayNormal)
~~10.~~*21.* BEFORE Notification Fail Time
22. RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (any valid process ID),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (E2),
'Time Stamp' = (any valid time stamp),
'Notification Class' = (the class corresponding to the object being tested),
'Priority' = (any valid value),
'Event Type' = (any valid event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = EVENT,
'AckRequired' = TRUE | FALSE,
'From State' = NORMAL,
'To State' = (any valid value),
'Event Values' = (values appropriate to the event type)
~~11.~~*23.* TRANSMIT BACnet-SimpleACK-PDU

7.3.1.17 Event_Message_Texts Tests

Reason For Change: Add new Notes to Tester to indicate how to execute the tests with algorithms that do not support the pTimeDelay or pTimeDelayNormal parameters. Fix spelling errors. Update test to match formatting of other tests, including updating to the new numbering schema.

Purpose: To verify that the value of the Event_Message_Texts property is updated when an object generates an event notification.

Test Concept: Read the Event_Message_Texts from the object. Transition the object through each event state which is enabled in the object saving the Message Text parameter from the received notification. Verify that the Event_Message_Texts updates with the Event_Message_Texts value received from the notification.

Configuration Requirements: The IUT shall be configured with an event-generation object, O1 which shall be in a NORMAL Event_State at the beginning of the test, and if Event_Enable is configurable, it shall have all bits set to TRUE for which the object supports ~~transtions.~~*transitions.*

Notes to Tester: For algorithms that do not apply the pTimeDelay or pTimeDelayNormal parameters, assume pTimeDelay or pTimeDelayNormal equals zero when executing the test.

Test Steps:
1. READ EMT = Event_Message_Texts
2. IF (Event_Enable is (TRUE, ?, ?) ~~and~~*AND* O1 can generate TO_OFFNORMAL transitions) THEN {
3. IF (pMonitoredValue is writable) THEN
4. WRITE pMonitoredValue = (a value that is offnormal)
ELSE
5. MAKE (pMonitoredValue a value that is offnormal)
6. WAIT (pTimeDelay)
7. BEFORE Notification Fail Time
8. RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (any valid process ID),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = O1,

© 2026 by BACnet International. All rights reserved.

'Time Stamp' = (any valid timestamp),
'Notification Class' = (the class corresponding to the object being tested),
'Priority' = (the configured TO_OFFNORMAL priority),
'Event Type' = (any valid event type),
'Message Text' = (M: any valid value placed into EMT[1]),
'Notify Type' = Notify_Type,
'AckRequired' = (the configured value for the TO_OFFNORMAL transition),
'From State' = NORMAL,
'To State' = (any valid offnormal state),
'Event Values' = (values appropriate to the event type)

9. VERIFY Event_Message_Texts = EMT

}

~~3~~*10.* IF (Event_Enable is (?, ?, TRUE) ~~and~~*AND* O1 can generate TO_NORMAL transitions) THEN {

11. IF (pMonitoredValue is writable) THEN

12. WRITE pMonitoredValue = (a value that will result in a TO_NORMAL transition)

ELSE

13. MAKE (pMonitoredValue a value that will result in a TO_NORMAL transition)

14. WAIT (pTimeDelayNormal)

15. BEFORE Notification Fail Time

16. RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (any valid process ID),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = O1,
'Time Stamp' = (any valid time stamp),
'Notification Class' = (the class corresponding to the object being tested),
'Priority' = (the configured TO_NORMAL priority),
'Event Type' = (any valid event type),
'Message Text' = (M: any valid value placed into EMT[3]),
'Notify Type' = Notify_Type,
'AckRequired' = (the configured value for the TO_NORMAL transition),
'From State' = (any valid value),
'To State' = NORMAL,
'Event Values' = (values appropriate to the event type)

17. VERIFY Event_Message_Texts = EMT

}

~~4~~*18.* IF (Event_Enable is (?, TRUE, ?) and O1 can generate TO_FAULT transitions) THEN {

19. MAKE (a condition exist that will cause O1 to generate a TO_FAULT transition)

20. BEFORE Notification Fail Time

21. RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (any valid process ID),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = O1,
'Time Stamp' = (any valid time stamp),
'Notification Class' = (the class corresponding to the object being tested),
'Priority' = (the configured TO_FAULT priority),
'Event Type' = (IF (Protocol_Revision < 13 THEN
(any valid event type),
ELSE
(CHANGE_OF_RELIABILITY,
'Message Text' = (M: any valid value placed into EMT[2]),
'Notify Type' = Notify_Type,
'AckRequired' = (the configured value for the TO_FAULT transition),
'From State' = (any valid value),
'To State' = FAULT,
'Event Values' = (values appropriate to the event type)

22. VERIFY Event_Message_Texts = EMT

© 2026 by BACnet International. All rights reserved.

}

7.3.1.18 Event_Message_Texts_Config Test

Reason For Change: Add new Notes to Tester to indicate how to execute the tests with algorithms that do not support the pTimeDelay or pTimeDelayNormal parameters. Move Notes to Tester at end of test to above test steps. Move note at end of Test Concept to Notes to Tester. Update test to match formatting of other tests, including updating to the new numbering schema.

Purpose: To verify that the Message Text parameter of generated event notifications is controlled via the Event_Message_Texts_Config property.

Test Concept: Select an object, O1, in the IUT that supports the Event_Message_Texts_Config property. Make O1 perform each supported event transition ~~(i.e., to-offnormal, to-normal, and to-fault)~~. Verify that the 'Message Text' parameter matches the associated Event_Message_Texts_Config value. ~~Note that due to the use of text substitution codes, the resulting text might not be an exact match.~~

Configuration Requirements: If possible, configure each entry in the Event_Message_Texts_Config property of Object O1 to be distinct. ES1 shall be the state to which O1 transitions. D1 is either the pTimeDelay, or pTimeDelayNormal parameter, or 0 (for transitions to and from FAULT state) depending on the event transition. ESINDEX shall be the array index associated with ES1 (1 for offnormal states, 2 for FAULT, and 3 for NORMAL). The notification class for O1 is configured for UnconfirmedEventNotification.

Notes to Tester: For algorithms that do not apply the pTimeDelay or pTimeDelayNormal parameters, assume D1 equals zero when executing the test.

Notes to Tester: Due to the use of text substitution codes, the resulting text might not be an exact match.

Notes to Tester: This behavior can alternately be tested using the ConfirmedEventNotification service, but it is not necessary to test both.

Test Steps:
1. MAKE (a condition exist which will cause O1 to transition to ES1)
2. WAIT D1
3. BEFORE Notification Fail Time
4. RECEIVE UnconfirmedEventNotification-Request
   'Process Identifier' = (any valid process identifier),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = O1,
   'Time Stamp' = (any valid time stamp),
   'Notification Class' = (the notification class configured for O1),
   'Priority' = (any valid priority),
   'Event Type' = (any standard event type),
   'Message Text' = T1,
   'Notify Type' = ALARM | EVENT,
   'AckRequired' = TRUE | FALSE,
   'From State' = (any valid event state),
   'To State' = ES1,
   'Event Values' = (any values appropriate to the event type)

~~4.~~*5.* CHECK (T1 is equivalent to Event_Message_Texts_Config[ESINDEX] with any text substitutions as defined by the vendor)

~~Notes to Tester: This behavior can alternately be tested using the ConfirmedEventNotification service, but it is not necessary to test both.~~

© 2026 by BACnet International. All rights reserved.

7.3.1.19 Event_Algorithm_Inhibit Tests

7.3.1.19.1 Event_Algorithm_Inhibit Test

Reason For Change: Add new Notes to Tester to indicate how to execute the tests with algorithms that do not support the pTimeDelay or pTimeDelayNormal parameters. Move Notes to Tester from after test to before test. Update test to match formatting of other tests, including updating to the new numbering schema. Fix Typo in Step 8.

Purpose: To verify that Event_Algorithm_Inhibit property in objects with intrinsic or algorithmic reporting controls whether or not the event state detection algorithm is executed.

Test Concept: Select an event generating object, O1, which supports the Event_Algorithm_Inhibit property and does not support the Event_Algorithm_Inhibit_Ref property or where the Event_Algorithm_Inhibit_Ref property is present and not initialized. With Event_Algorithm_Inhibit set to FALSE, make a condition exist that should result in an event transition to a ~~normal~~*NORMAL* or offnormal state. Verify that a transition occurs and that a notification is generated. Set Event_Algorithm_Inhibit to TRUE. If not already in a NORMAL state, verify that the object transitions to NORMAL. Make a condition exist that should result in an event transition, if the object Event_Algorithm_Inhibit were FALSE. If O1 supports fault detection, make a fault condition exist and verify that object detects the fault and transitions to FAULT.

Configuration Requirements: O1 is configured to detect and report unconfirmed events, is in the NORMAL state, and, if supported, is configured to detect fault conditions. D1 is either the pTimeDelay, or pTimeDelayNormal parameter, or 0 (for transitions to and from FAULT state) depending on the event transition.

Notes to Tester: For algorithms that do not apply the pTimeDelay or pTimeDelayNormal parameters, assume D1 equals zero when executing the test.

Notes to Tester: This behavior can alternately be tested using the ConfirmedEventNotification service, but it is not necessary to test both.

Test Steps:

1. VERIFY pCurrentState = NORMAL
2. VERIFY Event_Algorithm_Inhibit = FALSE
3. MAKE (a condition exist which will result in a transition of O1. If possible, 'To State' shall be an offnormal event state)
4. WAIT D1
5. BEFORE Notification Fail Time
6. RECEIVE UnconfirmedEventNotification-Request
   'Process Identifier' = (~~PID1: any valid process identifier~~*any valid process ID*),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = O1,
   'Time Stamp' = (any valid time stamp),
   'Notification Class' = (the notification class configured for O1),
   'Priority' = (the value configured for the transition),
   'Event Type' = (ET1: any valid event type),
   'Message Text' = (optional, any valid message text),
   'Notify Type' = (value from the Notify_Type property configured for O1),
   'AckRequired' = TRUE | FALSE,
   'From State' = NORMAL,
   'To State' = (ES1: any event state appropriate to the event type)
   'Event Values' = (any values appropriate to the event type)

~~6.~~*7.* WRITE Event_Algorithm_Inhibit = TRUE

~~7.~~*8.* IF (ES1 <> NORMAL) THEN
   BEFORE Notification Fail Time
   RECEIVE UnconfirmedEventNotification-Request

© 2026 by BACnet International. All rights reserved.

'Process Identifier' = ~~PID1~~*any valid process ID*,
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = O1,
'Time Stamp' = (any valid time stamp),
'Notification Class' = (the notification class configured for O1),
'Priority' = (the value configured for the transition),
~~'Event Type' ET1,~~*'Event Type' = ET1*
'Message Text' = (optional, any valid message text),
'Notify Type' = (value from the Notify_Type property configured for O1),
'AckRequired' = TRUE | FALSE,
'From State' = ES1,
'To State' = NORMAL,
'Event Values' = (any values appropriate to the event type)

~~8.~~*9.* VERIFY pCurrentState = NORMAL
~~9.~~*10.* MAKE (a condition exist which would result in a transition of O1 other than TO-FAULT, if Event_Algorithm_Inhibit were FALSE)
~~10.~~*11.* WAIT D1
~~11.~~*12.* WAIT Notification Fail Time
~~12.~~*13.* CHECK (that the IUT did not send any event notifications for O1)
~~13.~~*14.* VERIFY pCurrentState = NORMAL
~~14.~~*15.* IF (O1 supports fault detection) THEN {
16. MAKE (a fault condition exist for O1)
17. BEFORE Notification Fail Time
18. RECEIVE UnconfirmedEventNotification-Request
'Process Identifier' = ~~PID1~~*any valid process ID,*
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = O1,
'Time Stamp' = (any valid time stamp),
'Notification Class' = (the notification class configured for O1),
'Priority' = (the value configured for the transition),
'Event Type' = CHANGE_OF_RELIABILITY,
'Message Text' = (optional, any valid message text),
'Notify Type' = (value from the Notify_Type property configured for O1),
'AckRequired' = TRUE | FALSE,
'From State' = NORMAL,
'To State' = FAULT,
'Event Values' = (any values appropriate for CHANGE_OF_RELIABILITY)
19. MAKE (remove the fault condition)
20. WAIT (pTimeDelayNormal)
21. BEFORE Notification Fail Time
22. RECEIVE UnconfirmedEventNotification-Request
'Process Identifier' = ~~PID1~~*any valid process ID*,
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = O1,
'Time Stamp' = (any valid time stamp),
'Notification Class' = (the notification class configured for O1),
'Priority' = (the value configured for the transition),
'Event Type' = CHANGE_OF_RELIABILITY,
'Message Text' = (optional, any valid message text),
'Notify Type' = (value from the Notify_Type property configured for O1),
'AckRequired' = TRUE | FALSE,
'From State' = FAULT,
'To State' = NORMAL,
'Event Values' = (any values appropriate for CHANGE_OF_RELIABILITY)
}

© 2026 by BACnet International. All rights reserved.

~~Notes to Tester: This behavior can alternately be tested using the ConfirmedEventNotification service, but it is not necessary to test both.~~

7.3.1.21 Reliability_Evaluation_Inhibit Tests

7.3.1.21.1 Reliability_Evaluation_Inhibit Test

Reason for Change: Fixed step 20.

Purpose: To verify that Reliability_Evaluation_Inhibit controls whether or not fault conditions are detected and events are generated.

Test Concept: Select an event generating object, O1, which supports the Reliability_Evaluation_Inhibit property. With Reliability_Evaluation_Inhibit FALSE, make a fault condition exist. Verify that Reliability changes and that a notification is generated. Set Reliability_Evaluation_Inhibit to TRUE. Verify that the Reliability changes to NO_FAULT_DETECTED and that a TO_NORMAL notification is generated. Remove the fault condition and ensure that no notification is generated. Make a fault condition exist and verify that Reliability remains NO_FAULT_DETECTED, and that no notification is generated.

Configuration Requirements: O1 is configured to detect and report unconfirmed events, is in the NORMAL state, and Reliability_Evaluation_Inhibit equals FALSE, so that reliability evaluation for that object is configured to detect fault conditions.

Notes to Tester: This behavior can alternately be tested using the ConfirmedEventNotification service, but it is not necessary to test both.

Test Steps:

1. VERIFY pCurrentState = NORMAL
2. VERIFY Reliability = NO_FAULT_DETECTED
3. MAKE(~~a~~ condition exist that would cause O1 to generate a TO_FAULT transition)
4. BEFORE **Notification Fail Time**
5. RECEIVE UnconfirmedEventNotification-Request
   'Process Identifier' = (the value configured for the transition),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = O1,
   'Time Stamp' = (any valid timestamp),
   'Priority' = (any valid priority),
   'Event Type' = CHANGE_OF_RELIABILITY,
   'Message Text' = (optional, any valid message text),
   'Notify Type' = ALARM | EVENT,
   'AckRequired' = TRUE | FALSE,
   'From State' = NORMAL,
   'To State' = FAULT,
   'Event Values' = (any values appropriate to CHANGE_OF_RELIABILITY)
6. VERIFY Reliability <> NO_FAULT_DETECTED
7. VERIFY pCurrentState = FAULT
8. IF Reliability_Evaluation_Inhibit is writable THEN
9. WRITE Reliability_Evaluation_Inhibit = TRUE
   ELSE
10. MAKE(Reliability_Evaluation_Inhibit TRUE)
11. BEFORE **Internal Processing Fail Time** + **Notification Fail Time**
12. RECEIVE UnconfirmedEventNotification-Request
    'Process Identifier' = (the value configured for the transition),
    'Initiating Device Identifier' = IUT,

© 2026 by BACnet International. All rights reserved.

'Event Object Identifier' = O1,
'Time Stamp' = (any valid timestamp),
'Priority' = (any valid priority),
'Event Type' = CHANGE_OF_RELIABILITY,
'Message Text' = (optional, any valid message text),
'Notify Type' = ALARM | EVENT,
'AckRequired' = TRUE | FALSE,
'From State' = FAULT,
'To State' = NORMAL,
'Event Values' = (any values appropriate to CHANGE_OF_RELIABILITY)

13. VERIFY Reliability = NO_FAULT_DETECTED
14. VERIFY pCurrentState = NORMAL
15. MAKE(remove the fault condition)
16. WAIT(pTimeDelayNormal)
17. WAIT **Notification Fail Time**
18. CHECK (that the IUT did not send any event notifications for O1)
19. VERIFY Reliability = NO_FAULT_DETECTED
20. MAKE(~~O1~~a condition exist that would cause O1 to generate a TO_NORMAL transition)
21. WAIT **Notification Fail Time**
22. VERIFY Reliability = NO_FAULT_DETECTED
23. VERIFY pCurrentState = NORMAL
24. CHECK (that the IUT did not send any event notifications for O1)

7.3.1.21.3 Reliability_Evaluation_Inhibit Object Test

Reason for Change: Updating step 5 with condition, so that Devices having objects which do not support Event Reporting should be verified against correct state of Event.

Purpose: To verify that Reliability_Evaluation_Inhibit controls whether or not an object performs the reliability-evaluation process.

Test Concept: Select an object, O1, which supports the Reliability_Evaluation_Inhibit property. With Reliability_Evaluation_Inhibit FALSE, make a fault condition exist. Verify that the Reliability property changes. Set Reliability_Evaluation_Inhibit to TRUE. Verify that the Reliability changes to NO_FAULT_DETECTED. Make a fault condition exist and verify that Reliability property remains NO_FAULT_DETECTED.

Configuration Requirements: The Event_State of O1 is NORMAL, and Reliability_Evaluation_Inhibit equals FALSE.

Test Steps:

1. VERIFY Event_State = NORMAL
2. VERIFY Reliability = NO_FAULT_DETECTED
3. MAKE (a fault condition exist for O1)
4. VERIFY Reliability <> NO_FAULT_DETECTED
~~5. VERIFY Event_State = FAULT~~
5. IF (event reporting supported) THEN
6. VERIFY Event_State = FAULT
ELSE
7. VERIFY Event_State = NORMAL
~~6~~*8.* IF Reliability_Evaluation_Inhibit is writable THEN
~~7~~*9.* WRITE Reliability_Evaluation_Inhibit = TRUE
ELSE
~~8~~*10.* MAKE (Reliability_Evaluation_Inhibit TRUE)
~~9~~*11.*VERIFY Reliability = NO_FAULT_DETECTED
~~10~~*12.* VERIFY Event_State = NORMAL

© 2026 by BACnet International. All rights reserved.

~~11~~*13*. MAKE (remove the fault condition)
~~12~~*14*. VERIFY Reliability = NO_FAULT_DETECTED
~~13~~*15*. MAKE (a fault condition exist for O1)
~~14~~*16*. VERIFY Reliability = NO_FAULT_DETECTED
~~15~~*17*. VERIFY Event_State = NORMAL

7.3.1.22 Event_Detection_Enable Tests

7.3.1.22.1 Event_Detection_Enable Inhibits Event Generation

Reason for Change: Add Notes to Tester to indicate how to handle events that do not support pTimeDelay or pTimeDelayNormal. Move conditionality in original Configuration Requirements to be a separate Test Conditionality. Rewrite Configuration Requirements for better clarity.

Reason for Change: Updated purpose

Purpose: To verify that Event_Detection_Enable ~~enables and~~ disables event detection ~~in objects which are configured for event reporting~~

Test Concept: Select an event generating object, O1, that supports event reporting. Verify the Event_State is NORMAL and ~~the~~ Acked_Transitions, Event_Time_Stamps, and Event_Message_Texts are equal to their respective initial conditions, as mandated in the standard. Make a condition exist that would cause a transition if event reporting were enabled and observe that no notification messages are transmitted.

Test Conditionality: If Event_Detection_Enable cannot be set to FALSE, this test shall be skipped.

~~Configuration Requirements: O1 is configured with Event_Detection_Enable set to FALSE. If Event_Detection_Enable cannot be set to FALSE, this test shall be skipped. DELAY shall represent the time delay appropriate to the transition being tested (i.e. Time_Delay for TO_OFFNORMAL, 0 for TO_FAULT and FAULT to NORMAL transitions, and either Time_Delay or Time_Delay_Normal for TO_NORMAL). For this test, NO_TS equals a BACnetDateTime with all unspecified values, a BACnet Time with all unspecified values, or a sequence number of 0.~~
Configuration Requirements: O1 is configured with Event_Detection_Enable set to FALSE. DELAY is the pTimeDelay parameter, pTimeDelayNormal parameter, or zero (for transitions to and from FAULT state) depending on the event transition. NO_TS equals a BACnetDateTime with all unspecified values, a BACnetTime with all unspecified values, or a sequence number of zero.

Notes to Tester: For algorithms that do not apply the pTimeDelay or pTimeDelayNormal parameters, assume DELAY equals zero when executing the test.

Notes to Tester: This behavior can alternately be tested using the ConfirmedEventNotification service, but it is not necessary to test both.

Test Steps:

1. VERIFY Event_Detection_Enable = FALSE
2. VERIFY pCurrentState = NORMAL
3. VERIFY Acked_Transitions = (T,T,T)
4. VERIFY Event_Time_Stamps = [NO_TS, NO_TS, NO_TS]
5. IF (Event_Message_Texts property exists) THEN
6. VERIFY Event_Message_Texts = ['', '', '']
7. MAKE (a condition exist which would cause O1 to transition, if Event_Detection_Enable were TRUE)
8. WAIT (DELAY + **Notification Fail Time)**
9. CHECK (that the IUT did not send any event notifications for O1)
10. VERIFY pCurrentState = NORMAL
11. VERIFY Acked_Transitions = (T,T,T)

© 2026 by BACnet International. All rights reserved.

12. VERIFY Event_Time_Stamps = [NO_TS, NO_TS, NO_TS]
13. IF (Event_Message_Texts property exists) THEN
14. VERIFY Event_Message_Texts = ['', '', '']

7.3.1.27 Blink Warn Tests

7.3.1.27.1 Blink-Warn WARN Command Test

Reason for Change: Remove physical blink warn check.

Purpose: To verify the correct operation of the blink-warn WARN command.

Test Concept: Select an object O1 that supports blink-warn WARN command. Ensure O1 is not in egress mode and the specific properties have been configured to support blink-warn. Execute blink-warn WARN command by writing C1 to PROP_REF at a priority PTY1 of O1 and validate the specified blink-warn command functions correctly. Validate the Priority_Array value at priority PTY1 remains.

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. The Priority_Array at PTY1 has a value V1, Blink_Warn_Enable is TRUE, Egress_Active is FALSE.

	Binary Lighting Output object	**Lighting Output object**
PROP_REF	Present_Value	Present_Value or Lighting_Command
C1	WARN	-1.0 if PROP_REF = Present_Value, otherwise WARN
V1	ON	>1.0

Test Steps:

1. VERIFY Priority_ Array = V1, ARRAY INDEX = PTY1
2. VERIFY Blink_Warn_Enable = TRUE
3. VERIFY Egress_Active = FALSE
4. WRITE PROP_REF = C1, PRIORITY = PTY1
~~5. BEFORE **Internal Processing Fail Time**~~
~~CHECK (blink-warn occurred)~~
-- blink warn occurs
~~6~~*5*. VERIFY Egress_Active = FALSE
~~7~~*6*. VERIFY Priority_ Array = V1, ARRAY INDEX = PTY1

7.3.1.27.2 Blink-Warn WARN_OFF Command Test

Reason for Change: Remove physical blink warn check.

Purpose: To verify the correct operation of the blink-warn WARN_OFF command.

Test Concept: Select an object O1 that supports blink-warn commands. Ensure O1 is not in egress mode and the specific properties have been configured to support blink-warn. Execute blink-warn WARN_OFF command by writing C1 to PROP_REF at a priority PTY1 of O1 and validate the specified blink-warn command functions correctly. Validate the Priority_Array value at priority PTY1 after Egress_Time seconds has elapsed.

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. The Priority_Array at PTY1 has a value V1, Blink_Warn_Enable is TRUE, Egress_Active is FALSE, and Egress_Time is a non-zero value.

© 2026 by BACnet International. All rights reserved.

	Binary Lighting Output object	Lighting Output object
PROP_REF	Present_Value	Present_Value or Lighting_Command
C1	WARN_OFF	-3.0 if PROP_REF = Present_Value, otherwise WARN_OFF
V1	ON	>1.0
V2	OFF	0.0

Test Steps:

1. VERIFY Priority_ Array = V1, ARRAY INDEX = PTY1
2. VERIFY Blink_Warn_Enable = TRUE
3. VERIFY Egress_Time > 0
4. VERIFY Egress_Active = FALSE
5. WRITE PROP_REF = C1, PRIORITY = PTY1
6. *READ T1.date = Device, IUT, Local_Date*
7. *READ T1.time = Device, IUT, Local_Time*
~~6. T1 = current local time~~
~~7~~*8*. VERIFY Priority_Array = V1, ARRAY INDEX = PTY1
9. *WHILE (Egress_Active = TRUE) {}*
~~8. WHILE (Egress_Active = TRUE)~~
~~WAIT 1s~~
10. *READ T2.date = Device, IUT, Local_Date*
11. *READ T2.time = Device, IUT, Local_Time*
~~9. T2 = current local time~~
~~10. CHECK (blink warn occurred)~~
-- blink warn occurred
~~11~~*12*. VERIFY Egress_Time ~= (T2 – T1)
~~12~~*13*. VERIFY Priority_ Array = V2, ARRAY INDEX = PTY1

7.3.1.27.3 Blink-Warn WARN_RELINQUISH Command Test

Reason for Change: Remove physical blink warn check.

Purpose: To verify the correct operation of the blink-warn WARN_RELINQUISH commands.

Test Concept: Select an object O1 that supports blink-warn commands. Ensure O1 is not in egress mode and the specific properties have been configured to support blink-warn. Execute blink-warn WARN_RELINQUISH command by writing C1 to PROP_REF at a priority PTY1 of O1 and validate the specified blink-warn command functions correctly. Validate the Priority_Array value at priority PTY1 after Egress_Time seconds has elapsed.

Configuration Requirements: O1 shall be configured such that all slots in the Priority_Array numerically less than PTY1 shall have a value of NULL, at least one slot numerically greater than PTY1 or Relinquish_Default shall have a value of V2, and all other slots numerically greater than PTY1 shall have a value of V0. No internal algorithms are issuing commands to O1 at any priority. The Priority_Array at PTY1 has a value V1, Blink_Warn_Enable is TRUE, Egress_Time is a non-zero value, and Egress_Active is FALSE.

	Binary Lighting Output object	Lighting Output object
PROP_REF	Present_Value	Present_Value or Lighting_Command
C1	WARN_RELINQUISH	-2.0 if PROP_REF = Present_Value, otherwise WARN_OFF
V0	NULL or OFF	NULL or 0.0
V1	ON	>1.0
V2	OFF	0.0

Test Steps:

1. VERIFY Priority_ Array = V1, ARRAY INDEX = PTY1

© 2026 by BACnet International. All rights reserved.

2. VERIFY Blink_Warn_Enable = TRUE
3. VERIFY Egress_Time > 0
4. VERIFY Egress_Active = FALSE
5. WRITE PROP_REF = C1, PRIORITY = PTY1
6. READ T1.date = Device, IUT, Local_Date
7. READ T1.time = Device, IUT, Local_Time
~~6. T1 = current local time~~
~~7~~*8*. VERIFY Priority_Array = V1, ARRAY INDEX = PTY1
9. WHILE (Egress_Active = TRUE) {}
-- blink warn occurred
~~8. WHILE (Egress_Active = TRUE)~~
~~WAIT 1s~~
~~9. T2 = current local time~~
~~10. CHECK (blink warn occurred)~~
10. READ T2.date = Device, IUT, Local_Date
11. READ T2.time = Device, IUT, Local_Time
~~11~~*12*. VERIFY Egress_Time ~= (T2 – T1)
~~12~~*13*. VERIFY Priority_ Array = NULL, ARRAY INDEX = PTY1

7.3.1.27.4 Blink-Warn STOP Command Test

Reason for Change: Remove physical blink warn check.

Purpose: To verify the correct operation of the blink-warn STOP command.

Test Concept: Select an object O1 that supports blink-warn commands. Ensure O1 is not in egress mode and the specific properties have been configured to support blink-warn. Execute blink-warn command by writing C1 to PROP_REF at a priority PTY1 of O1 *causing a blink-warn to occur* ~~and validate that blink-warn occurs~~. Before the Egress_Time times out, STOP the egress process and validate the Priority_Array value at PTY1 remains equal to V1 after Egress_Time.

Configuration Requirements: O1 shall be configured such that all slots in the Priority_Array numerically less than PTY1 shall have a value of NULL, at least one slot numerically greater than PTY1 or Relinquish_Default shall have a value of V2, and all other slots numerically greater than PTY1 shall have a value of V0. No internal algorithms are issuing commands to O1 at a priority numerically less than or equal to PTY1. The Priority_Array at PTY1 has a value V1, Blink_Warn_Enable is TRUE, Egress_Time is a non-zero value, and Egress_Active is FALSE.

	Binary Lighting Output object	**Lighting Output object**
PROP_REF	Present_Value	Lighting_Command
C1	WARN_RELINQUISH or WARN_OFF	WARN_RELINQUISH or WARN_OFF
V0	NULL or OFF	NULL or 0.0
V1	ON	>1.0
V2	OFF	0.0

Test Steps:

1. READ V1 = Priority_ Array, ARRAY INDEX = PTY1
~~1. VERIFY Priority_ Array = V1, ARRAY INDEX = PTY1~~
2. VERIFY Blink_Warn_Enable = TRUE
3. VERIFY Egress_Time > 0
4. VERIFY Egress_Active = FALSE
5. WRITE PROP_REF = C1, PRIORITY = PTY1
~~6. T1 = current local time~~
~~7. BEFORE **Internal Processing Fail Time**~~
--(blink-warn occurs)

© 2026 by BACnet International. All rights reserved.

~~CHECK (blink-warn occurred)~~
~~8~~6. VERIFY Egress_Active = TRUE
7. WRITE PROP_REF = STOP, PRIORITY = PTY1
~~9. WAIT less than Egress_Time~~
~~WRITE PROP_REF = STOP, PRIORITY = PTY1~~
~~10. T2 = current local time~~
~~11. WAIT **Internal Processing Fail Time**~~
~~12~~*8*.VERIFY Egress_Active = FALSE
~~13. WAIT Egress_Time + (T2 – T1) + **Internal Processing Fail Time**~~
~~14~~*9*.VERIFY Priority_ Array = V1, ARRAY INDEX = PTY1

7.3.1.27.5 Blink-Warn WARN Command Failure Test

Reason for Change: Remove physical blink warn check.

Purpose: To verify blink-warn WARN command does not occur when, the specified priority is not the highest active priority, the value at the specified priority is off or Blink_Warn_Enable is FALSE.

Test Concept: Select an object O1 that supports blink-warn commands. Configure O1 such that a blink-warn command would generate a blink-warn except set the specified failure conditions. Verify blink-warn does not occur and the Priority_Array is not affected.

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. Select a priority, PTY2, which is numerically less than PTY1 and not equal to 6. Blink_Warn_Enable is TRUE, Egress_Active is FALSE.

	Binary Lighting Output object	**Lighting Output object**
PROP_REF	Present_Value	Present_Value or Lighting_Command
C1	WARN	-1.0 if PROP_REF = Present_Value, otherwise WARN
V1, V2	ON	>1.0
V3	OFF	0.0

Test Steps:

-- Test for the specified priority is not the highest active priority
1. VERIFY Blink_Warn_Enable = TRUE
2. WRITE Present_Value = V1, PRIORITY = PTY1
3. VERIFY Egress_Active = FALSE
4. WRITE Present_Value = V2, PRIORITY = PTY2
5. WRITE PROP_REF = C1, PRIORITY = PTY1
~~6. WAIT **Internal Processing Fail Time**~~
~~CHECK (blink-warn did not occur)~~
-- blink warn does not occur
~~7~~*6*. VERIFY Egress_Active = FALSE
~~8~~*7*. VERIFY Priority_Array = V1, ARRAY INDEX = PTY1
~~9~~*8*. WRITE Present_Value = NULL, PRIORITY = PTY2

-- Test for the value at the specified priority is either OFF or 0.0
~~10~~*9*.WRITE Present_Value = V3, PRIORITY = PTY1
~~11~~*10*. WRITE PROP_REF = C1, PRIORITY = PTY1
~~12. WAIT **Internal Processing Fail Time**~~
~~CHECK (blink-warn did not occur)~~
-- blink warn does not occur
~~13~~*11*. VERIFY Egress_Active = FALSE
~~14~~*12*. VERIFY Priority_ Array = V3, ARRAY INDEX = PTY1

© 2026 by BACnet International. All rights reserved.

~~15~~*13*. WRITE Present_Value = V1, PRIORITY = PTY1

-- Test for Blink_Warn_Enable is FALSE
~~16~~*14*. IF (Blink_Warn_Enable is writable) THEN
15. WRITE Blink_Warn_Enable = FALSE
16. WRITE PROP_REF = C1, PRIORITY = PTY1
~~WAIT **Internal Processing Fail Time**~~
~~CHECK (blink-warn did not occur)~~
-- blink warn does not occur
17. VERIFY Egress_Active = FALSE
18. VERIFY Priority_ Array = V1, ARRAY INDEX = PTY1

7.3.1.27.6 Blink-Warn WARN_OFF Command Failure Test

Reason for Change: Remove physical blink warn check.

Purpose: To verify blink-warn WARN_OFF command does not occur when the specified priority is not the highest active priority, the Present_Value is either 0.0 or OFF, or Blink_Warn_Enable is FALSE.

Test Concept: Select an object O1 that supports blink-warn commands. Configure O1 such that a blink-warn command would generate a blink-warn except set the specified failure conditions. Verify blink-warn does not occur and the Priority_Array is correctly changed.

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. Blink_Warn_Enable is TRUE, Egress_Time is a non-zero value and Egress_Active is FALSE.

	Binary Lighting Output object	**Lighting Output object**
PROP_REF	Present_Value	Present_Value or Lighting_Command
C1	WARN_OFF	-3.0 if PROP_REF = Present_Value, otherwise WARN_OFF
V1, V2	ON	>1.0
V3	OFF	0.0

Test Steps:

-- Test for the specified priority is not the highest active priority
1. VERIFY Blink_Warn_Enable = TRUE
2. VERIFY Egress_Time > 0
3. WRITE Present_Value = V1, PRIORITY = PTY1
4. VERIFY Egress_Active = FALSE
5. WRITE Present_Value = V2, PRIORITY = PTY2, a value not equal to 6 and less than PTY1
6. WRITE PROP_REF = C1, PRIORITY = PTY1
~~7. WAIT **Internal Processing Fail Time**~~
~~CHECK (blink-warn did not occur)~~
-- blink warn does not occur
~~8~~*7*. VERIFY Egress_Active = FALSE
~~9~~*8*. VERIFY Priority_ Array = V3, ARRAY INDEX = PTY1
~~10~~*9*.WRITE Present_Value = V1, PRIORITY = PTY1

-- Test for the Present_Value is OFF or 0.0
~~11~~*10*. WRITE Present_Value = V3, PRIORITY = PTY2, a value not equal to 6 and less than PTY1
~~12~~*11*. WRITE PROP_REF = C1, PRIORITY = PTY1
~~13. WAIT **Internal Processing Fail Time**~~
~~CHECK (blink-warn did not occur)~~
-- blink warn does not occur
~~14~~*12*. VERIFY Egress_Active = FALSE

© 2026 by BACnet International. All rights reserved.

~~15~~*13*. VERIFY Priority_ Array = V3, ARRAY INDEX = PTY1
~~16~~*14*. WRITE Present_Value = NULL, PRIORITY = PTY2
~~17~~*15*. WRITE Present_Value = V1, PRIORITY = PTY1

-- Test for Blink_Warn_Enable is FALSE
~~18~~*16*. IF (Blink_Warn_Enable is writable) THEN
17. WRITE Blink_Warn_Enable = FALSE
18. WRITE PROP_REF = C1, PRIORITY = PTY1
~~WAIT **Internal Processing Fail Time**~~
~~CHECK (blink-warn did not occur)~~
-- *blink warn does not occur*
19. VERIFY Egress_Active = FALSE
20. VERIFY Priority_ Array = V3, ARRAY INDEX = PTY1

7.3.1.27.7 Blink-Warn WARN_RELINQUISH Command Failure Test

Reason for Change: Remove physical blink warn check.

Purpose: To verify blink-warn WARN_RELINQUISH command does not occur when the specified priority is not the highest active priority, the value at the specified priority is V0, the value of the next highest non-NULL priority, including Relinquish_Default, is V1, or Blink_Warn_Enable is FALSE.

Test Concept: Select an object O1 that supports blink-warn commands. Configure O1 such that a blink-warn command would generate a blink-warn except set the specified failure conditions. Verify blink-warn does not occur and the Priority_Array is correctly changed.

Configuration Requirements: O1 shall be configured such that all slots in the Priority_Array numerically less than PTY1 shall have a value of NULL, at least one slot numerically greater than PTY1 or Relinquish_Default shall have a value of V2, and all other slots numerically greater than PTY1 shall have a value of V0. No internal algorithms are issuing commands to O1 at any priority. Blink_Warn_Enable is TRUE, Egress_Time is a non-zero value, Egress_Active is FALSE.

	Binary Lighting Output object	**Lighting Output object**
PROP_REF	Present_Value	Present_Value or Lighting_Command
C1	WARN_RELINQUISH	-2.0 if PROP_REF = Present_Value, otherwise WARN_RELINQUISH
V0	OFF or NULL	0.0 or NULL
V1	ON	>1.0
V2	OFF	0.0

Test Steps:

-- Test for the specified priority is not the highest active priority
1. VERIFY Blink_Warn_Enable = TRUE
2. VERIFY Egress_Time > 0
3. WRITE Present_Value = V1, PRIORITY = PTY1
4. VERIFY Egress_Active = FALSE
5. WRITE Present_Value = V1, PRIORITY = PTY2, a value not equal to 6 and less than PTY1
6. WRITE PROP_REF = C1, PRIORITY = PTY1
~~7. WAIT **Internal Processing Fail Time**~~
~~CHECK (blink-warn did not occur)~~
-- *blink warn does not occur*
~~8~~*7*. VERIFY Egress_Active = FALSE
~~9~~*8*. VERIFY Priority_ Array = NULL, ARRAY INDEX = PTY1
~~10~~*9*.WRITE Present_Value = NULL, PRIORITY = PTY2

-- Test for the value at the specified priority is OFF or 0.0

© 2026 by BACnet International. All rights reserved.

~~11~~*10*. WRITE Present_Value = V2 PRIORITY = PTY1
~~12~~*11*. WRITE PROP_REF = C1, PRIORITY = PTY1
~~13. WAIT **Internal Processing Fail Time**~~
~~CHECK (blink-warn did not occur)~~
-- blink warn does not occur
~~14~~*12*. VERIFY Egress_Active = FALSE
~~15~~*13*. VERIFY Priority_ Array = NULL, ARRAY INDEX = PTY1

-- Test for the value at the specified priority is NULL
~~16~~*14*. WRITE Present_Value = NULL, PRIORITY = PTY1
~~17~~*15*. WRITE PROP_REF = C1, PRIORITY = PTY1
~~18. WAIT **Internal Processing Fail Time**~~
~~CHECK (blink-warn did not occur)~~
-- blink warn does not occur
~~19~~*16*. VERIFY Egress_Active = FALSE
~~20~~*17*. VERIFY Priority_ Array = NULL, ARRAY INDEX = PTY1

-- Test for the value of the next highest non-NULL priority is ON or > 1.0
~~21~~*18*. WRITE Present_Value = V1 PRIORITY = PTY1
~~22~~*19*. WRITE Present_Value = V1, PRIORITY = PTY3, a value numerically greater than PTY1
~~23~~*20*. WRITE PROP_REF = C1, PRIORITY = PTY1
~~24. WAIT **Internal Processing Fail Time**~~
~~CHECK (blink-warn did not occur)~~
-- blink warn does not occur
~~25~~*21*. VERIFY Egress_Active = FALSE
~~26~~*22*. VERIFY Priority_ Array = NULL, ARRAY INDEX = PTY1
~~27~~*23*. WRITE Present_Value = NULL, PRIORITY = PTY3

-- Test for the value of Relinquish_Default is ON or > 1.0
~~28~~*24*. IF (Relinquish_Default is writable) THEN
25. WRITE Present_Value = V1, PRIORITY = PTY1
26. WRITE Relinquish_Default = V1
27. WRITE PROP_REF = C1, PRIORITY = PTY1
~~WAIT **Internal Processing Fail Time**~~
~~CHECK (blink-warn did not occur)~~
-- blink warn does not occur
28. VERIFY Egress_Active = FALSE
29. VERIFY Priority_ Array = NULL, ARRAY INDEX = PTY1
30. WRITE Relinquish_Default = V2

-- Test for Blink_Warn_Enable is FALSE
~~29~~*31*. IF (Blink_Warn_Enable is writable) THEN
32. WRITE Present_Value = V1, PRIORITY = PTY1
33. WRITE Blink_Warn_Enable = FALSE
34. WRITE PROP_REF = C1, PRIORITY = PTY1
~~WAIT **Internal Processing Fail Time**~~
~~CHECK (blink-warn did not occur)~~
-- blink warn does not occur
35. VERIFY Egress_Active = FALSE
36. VERIFY Priority_ Array = NULL, ARRAY INDEX = PTY1

7.3.1.27.8 Blink-Warn WARN_OFF Command Halted Test

Reason for Change: Remove physical blink warn check.

© 2026 by BACnet International. All rights reserved.

Purpose: To verify blink-warn WARN_OFF execution is halted when a higher priority entry is written or the Present_Value at the specified priority is changed.

Test Concept: Select an object O1 that supports blink-warn commands. Configure O1 such that a blink-warn command will generate a blink-warn. Before the Egress timer expires, verify the specified actions clear the blink-warn properties and the Priority_Array is correctly changed.

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. Blink_Warn_Enable is TRUE, Egress_Time is a non-zero value and Egress_Active is FALSE.

	Binary Lighting Output object	**Lighting Output object**
PROP_REF	Present_Value	Present_Value or Lighting_Command
C1	WARN_OFF	-3.0 if PROP_REF = Present_Value, otherwise WARN_OFF
V1 to V3	ON	>1.0
V4	OFF	0.0

Test Steps:

-- Test for a higher priority entry is written to a non NULL value
1. WRITE Present_Value = V1, PRIORITY = PTY1
2. VERIFY Blink_Warn_Enable = TRUE
3. VERIFY Egress_Time > 0
4. VERIFY Egress_Active = FALSE
5. WRITE PROP_REF = C1, PRIORITY = PTY1
~~6. BEFORE **Internal Processing Fail Time**~~
~~CHECK (blink-warn occurred)~~
-- blink warn occurs
6. VERIFY Egress_Active = TRUE
7. WRITE Present_Value = V2, PRIORITY = PTY2, a value not equal to 6 and less than PTY1
~~7. BEFORE Egress_Active = FALSE~~
~~WRITE Present_Value = V2, PRIORITY = PTY2, a value not equal to 6 and less than PTY1~~
8. VERIFY Egress_Active = FALSE
9. VERIFY Priority_ Array = V4, ARRAY INDEX = PTY1
10. WRITE Present_Value = NULL, PRIORITY = PTY2

-- Test for the Present_Value at the specified property is changed
11. WRITE Present_Value = V1, PRIORITY = PTY1
~~12. VERIFY Blink_Warn_Enable = TRUE~~
~~13. VERIFY Egress_Time > 0~~
~~14. VERIFY Egress_Active = FALSE~~
~~15~~*12*. WRITE PROP_REF = C1, PRIORITY = PTY1
~~16. BEFORE **Internal Processing Fail Time**~~
~~CHECK (blink-warn occurred)~~
-- blink warn occurs
~~17. BEFORE Egress_Active = FALSE~~
~~WRITE Present_Value = V3, PRIORITY = PTY1~~
13. VERIFY Egress_Active = TRUE
14. WRITE Present_Value = V3, PRIORITY = PTY1
~~18~~*15*. VERIFY Egress_Active = FALSE
~~19~~*16*. VERIFY Priority_ Array = V3, ARRAY INDEX = PTY1

© 2026 by BACnet International. All rights reserved.

7.3.1.27.9 Blink-Warn WARN_RELINQUISH Command Halted Test

Reason for Change: Remove physical blink warn check.

Purpose: To verify blink-warn WARN_RELINQUISH execution is halted when a higher priority entry is written or the Present_Value at the specified priority is changed.

Test Concept: Select an object O1 that supports blink-warn commands. Configure O1 such that a blink-warn command will generate a blink-warn. Before the Egress timer expires, verify the specified actions clear the blink-warn properties and the Priority_Array is correctly changed.

Configuration Requirements: O1 shall be configured such that all slots in the Priority_Array numerically less than PTY1 have a value of NULL and at least one slot numerically greater than PTY1 or Relinquish_Default shall have a value of V1 and all other slots numerically greater than PTY1 shall have a value of V0. No internal algorithms are issuing commands to O1 at any priority. Blink_Warn_Enable is TRUE, Egress_Time is a non-zero value, Egress_Active is FALSE.

	Binary Lighting Output object	**Lighting Output object**
PROP_REF	Present_Value	Present_Value or Lighting_Command
C1	WARN_RELINQUISH	-2.0 if PROP_REF = Present_Value, otherwise WARN_RELINQUISH
V0	OFF or NULL	0.0 or NULL
V1	OFF	0.0
V2	ON	>1.0
V3	OFF	any value >1.0 and not equal to V2

Test Steps:

-- Test for a higher priority entry is written to a non NULL value
1. WRITE Present_Value = V2, PRIORITY = PTY1
2. VERIFY Blink_Warn_Enable = TRUE
3. VERIFY Egress_Time > 0
4. VERIFY Egress_Active = FALSE
5. WRITE PROP_REF = C1, PRIORITY = PTY1
~~6. BEFORE **Internal Processing Fail Time**~~
~~CHECK (blink-warn occurred)~~
-- blink warn occurs
6. VERIFY Egress_Active = TRUE
~~7. BEFORE Egress_Active = FALSE~~
~~WRITE Present_Value = V2, PRIORITY = PTY2, a value not equal to 6 and less than PTY1~~
7. WRITE Present_Value = V2, PRIORITY = PTY2, a value not equal to 6 and less than PTY1
8. VERIFY Egress_Active = FALSE
9. VERIFY Priority_ Array = NULL, ARRAY INDEX = PTY1
10. WRITE Present_Value = NULL, PRIORITY = PTY2

-- Test for the Present_Value at the specified property is changed
11. WRITE Present_Value = V2, PRIORITY = PTY1
~~12. VERIFY Blink_Warn_Enable = TRUE~~
~~13. VERIFY Egress_Time > 0~~
~~14. VERIFY Egress_Active = FALSE~~
~~15~~*12.* WRITE PROP_REF = C1, PRIORITY = PTY1
~~16. BEFORE **Internal Processing Fail Time**~~
~~CHECK (blink-warn occurred)~~
-- blink warn occurs
13. VERIFY Egress_Active = TRUE
~~17. BEFORE Egress_Active = FALSE~~
~~WRITE Present_Value = V3, PRIORITY = PTY1~~

© 2026 by BACnet International. All rights reserved.

14. WRITE Present_Value = V3, PRIORITY = PTY1
~~18~~*15.* VERIFY Egress_Active = FALSE
~~19. VERIFY Priority_Array = V3, ARRAY INDEX = PTY1~~

7.3.1.28 Value Source Mechanism Tests

7.3.1.28.1 Writing to the Value_Source Property by a Device Other than the Device that Commanded the Object

Reason for Change: Test does not check the value source properties for a commandable object. PRIO is not defined.

Purpose: To verify the IUT correctly refuses an attempt to write a Value_Source property by a device other than the device that most recently commanded the object.

Test Concept: Command an object, O1, that supports the value source mechanism, from device D1 *(TD)*, and verify the updated Value_Source *and, if commandable, the other value source properties*. Attempt to write to the Value_Source property from device D2. Verify that an error is returned and Value_Source *and the other value source properties do*~~does~~ not change.

Configuration Requirements: If commandable, the value selected for PRIO shall be numerically small enough to be the active priority.

Notes to Tester: The value of the network-number in the network address of the Value_Source property will be one of three values. It will be 0 if the TD and IUT are on the same network and the IUT does not know the network number.
It will be the network number of the local network if the IUT knows the network number and the TD and IUT are on the same network. It will be the TD's network number if the TD is on a different network.

Test Steps:

1. TRANSMIT WriteProperty-Request,
   SOURCE = D1,
   'Object Identifier' = O1,
   'Property Identifier' = (P1: the property monitored by the Value_Source mechanism for this object type),
   'Priority' = (*absent or* PRIO~~: absent or any value other than 6~~)
   'Property Value' = (X2: any valid value)
2. RECEIVE BACnet-SimpleACK-PDU
3. VERIFY Value_Source = (VS, D1's device identifier or network address)
~~3~~*4.* IF (O1 is commandable) THEN
5. VERIFY Priority_Array = X2, ARRAY_INDEX = PRIO
6. VERIFY Value_Source_Array = VS, ARRAY_INDEX = PRIO
7. VERIFY Last_Command_Time ~= (T1, the current local time)
8. IF (Command_Time_Array is present) THEN
9. VERIFY Command_Time_Array = T1, ARRAY_INDEX = PRIO
   ELSE
10. VERIFY (P1) = X2
~~4. VERIFY Value_Source = (D1's device identifier or network address)~~
-- Attempt to write D1's device identifier from D2
~~5~~*11.*TRANSMIT WriteProperty-Request,
   SOURCE = D2,
   'Object Identifier' = O1,
   'Property Identifier' = Value_Source,
   'Priority' = PRIO,
   'Property Value' = (any valid value)
~~6~~*12.*RECEIVE BACnet-Error PDU,
   'Error Class' = PROPERTY,
   'Error Code' = WRITE_ACCESS_DENIED
-- Verify value source properties are unchanged

© 2026 by BACnet International. All rights reserved.

~~7~~*13.* VERIFY Value_Source = (*VS*~~D1's device identifier or network address~~)
14. IF (O1 is commandable) THEN
15. VERIFY Value_Source_Array = VS, ARRAY_INDEX = PRIO
16. VERIFY Last_Command_Time = T1
17. IF (Command_Time_Array is present) THEN
18. VERIFY Command_Time_Array = T1, ARRAY_INDEX = PRIO

7.3.1.28.2 Non-commandable Value_Source Property Test

Reason for Change: Specify what is supposed to be written to the Value_Source property.

Purpose: To verify that the Value_Source property indicates the source of the current Present_Value in a non-commandable object.

Test Concept: Select a non-commandable object with a writable Present_Value which supports the Value Source mechanism. *The TD writes the* Present_Value ~~is written~~, and it is verified that Value_Source is updated appropriately. Value_Source is then written to verify that the last writer*, TD,* can update it.

Notes to Tester: The value of the network-number in the network address of the Value_Source property will be one of three values. It will be 0 if the TD and IUT are on the same network and the IUT does not know the network number.
It will be the network number of the local network if the IUT knows the network number and the TD and IUT are on the same network. It will be the TD's network number if the TD is on a different network.

Test Steps:

1. WRITE Present_Value = *(*V1*, any valid value)*
2. VERIFY Present_Value = V1
3. VERIFY Value_Source = (TD's device identifier or network address)
4. WRITE Value_Source = (*VS(TD's device identifier, O1, any valid object identifier*~~any valid value, V2~~*))*
5. VERIFY Value_Source = *VS*~~V2~~

7.3.1.28.3 Value_Source Property None Test

Reason for Change: The other Value Source properties are not tested. The minimum on/off requirement from 19.5.1.3 paragraph 4 is not tested. Allow Command_Time_Array to approximately equal T1.

Purpose: To verify that the Value_Source property shall have the value 'None' when there is no active value source.

Test Concept: If there is no active value source, i.e., the Present_Value has taken on the value of Relinquish_Default, then the Value_Source property shall have the value 'None' *and the other value source properties are updated. If minimum on/off is supported verify the Value_Source_Array at priority 6 is commanded object, O1, and the Command_Time_Array has the present local time.*

Configuration Requirements: The object (O1) to be tested shall have 1 non-NULL entry in its Priority_Array and the Current_Command_Priority has a value other than NULL or 6.

Test Steps:

1. READ PRIO = Current_Command_Priority
~~2. CHECK(PRIO <> 6 and PRIO <> NULL)~~
~~3~~*2*. VERIFY Value_Source = (is not 'None')
~~4~~*3*. WRITE Present_Value = NULL, PRIORITY = PRIO
~~5~~*4*. VERIFY Last_Command_Time ~= (*T1*, the current local time)
5. VERIFY Value_Source_Array = TD, ARRAY_INDEX = PRIO

© 2026 by BACnet International. All rights reserved.

6. IF (Command_Time_Array is present) THEN
7. VERIFY Command_Time_Array ~= T1, ARRAY_INDEX = PRIO
~~6~~*8*. IF (O1 has Minimum_On_Time or Minimum_Off_Time properties) THEN
9. WAIT the larger of Minimum_Off_Time and Minimum_On_Time
10. VERIFY Value_Source_Array = O1, ARRAY_INDEX = 6
11. IF (Command_Time_Array is present) THEN
12. VERIFY Command_Time_Array ~= (the current local time), ARRAY_INDEX = 6
~~7~~*13*.VERIFY Current_Command_Priority = NULL
~~8~~*14*.VERIFY Value_Source = 'None' -- the value is the choice 'none'

7.3.1.28.4 Commandable Value Source Test

Reason for Change: Command_Time_Array was not tested. Specify what is supposed to be written to the Value_Source property.

Purpose: To verify that the Value_Source, Value_Source_Array, and Last_Command_Time update correctly when Present_Value is written in a commandable object.

Test Concept: A commandable object which supports the value source mechanism is selected for the test. The Present_Value is written. Last_Command_Time, Value_Source, ~~and~~ Value_Source_Array, *and Command_Time_Array* properties are checked to verify that they have been updated appropriately. Value_Source is then written, and it is verified that Last_Command_Time property has not changed.

Configuration Requirements: The object being tested shall be commandable and support the Value Source mechanism. *The value selected for PRIO shall be numerically small enough to be the active priority.* ~~No other internal processes shall be controlling the object.~~

Notes to Tester: The value of the network-number in the network address of the Value_Source property will be one of three values. It will be 0 if the TD and IUT are on the same network and the IUT does not know the network number.
It will be the network number of the local network if the IUT knows the network number and the TD and IUT are on the same network. It will be the TD's network number if the TD is on a different network.

Test Steps:

-- Verify that the value source properties are updated when Present_Value is commanded.
1. WRITE Present_Value = (V1: any valid value), PRIORITY = (PRIO~~: any value other than 6~~)
2. VERIFY Value_Source_Array = (*VS*~~SRC1~~: TD's device identifier or network address), ARRAY_INDEX = PRIO
3. VERIFY Value_Source = *VS*~~SRC1~~
4. VERIFY Last_Command_Time ~= (*T1,* the current local time)
5. IF (Command_Time_Array is present) THEN
6. VERIFY Command_Time_Array = T1, ARRAY_INDEX = PRIO
-- Verify that Value_Source can be written and that Last_Command_Time does not update.

~~5. READ T1 = (O1), Last_Command_Time~~
~~6~~*7*. WRITE Value_Source = (*VS(TD's device identifier, O1, any valid object identifier*~~ SRC2: any valid value~~), PRIORITY = PRIO
8. VERIFY Value_Source = VS
~~7~~*9*. VERIFY Value_Source_Array = *VS*~~ SRC2~~, ARRAY_INDEX = PRIO
~~8. IF (Current_Command_Priority == PRIO) THEN~~
~~VERIFY Value_Source = SRC2~~
~~9~~*10*.VERIFY Last_Command_Time = T1

© 2026 by BACnet International. All rights reserved.

7.3.1.28.5 Life Safety Value_Source Property Test

Reason for Change: Specify what is supposed to be written to the Value_Source property.

Purpose: To verify that the Value_Source property indicates the source of the current Mode property in a life safety object.

Test Concept: Select a life safety object which supports the Value Source mechanism. Mode is written, and it is verified that Value_Source is updated appropriately. Value_Source is then written to verify that the last writer can update it.

Notes to Tester: The value of the network-number in the network address of the Value_Source property will be one of three values. It will be 0 if the TD and IUT are on the same network and the IUT does not know the network number.
It will be the network number of the local network if the IUT knows the network number and the TD and IUT are on the same network. It will be the TD's network number if the TD is on a different network.

Test Steps:

1. WRITE Mode = V1
2. VERIFY Mode = V1
3. VERIFY Value_Source = (TD's device identifier or network address)
4. WRITE Value_Source = (*VS(TD's device identifier, O1, any valid object identifier* ~~any valid value, V2~~)
5. VERIFY Value_Source = *VS* ~~V2~~

7.3.1.28.X1 Value Source Initiated Locally Test

Reason for Change: No test for this functionality.

Purpose: To verify that a change to the Present_Value initiated by the local device results in updates to the Value Source properties. This change to the Present_Value could be from a Program object, Schedule object or some mechanism internal to the device.

Test Concept: The Present_Value is changed by a mechanism local to the device. The Value_Source is verified and, if the object is commandable, Last_Command_Time, Value_Source_Array, and Command_Time_Array properties are checked to verify that they have been updated appropriately.

Configuration Requirements: If commandable, the value selected for PRIO shall be numerically small enough to be the active priority.

Test Steps:

1. MAKE (O1, Present_Value change by some mechanism local to the device. If the object is commandable, at a PRIORITY = PRIO)
2. IF (the mechanism is from an object, O2) THEN
3. VERIFY Value_Source = VS, (IUT's device identifier, O2)
   ELSE
4. VERIFY Value_Source = VS, (IUT's device identifier)
5. IF (O1 is commandable) THEN
6. VERIFY Value_Source_Array = VS, ARRAY_INDEX = PRIO
7. VERIFY Last_Command_Time ~= (T1, the current local time)
8. IF (Command_Time_Array is present) THEN
9. VERIFY Command_Time_Array = T1, ARRAY_INDEX = PRIO

© 2026 by BACnet International. All rights reserved.

7.3.1.X73 Number_Of_States Property Tests

7.3.1.X73.1 Writable Number_Of_States Test

Reason for Change: New Test

Purpose: This test verifies that when the value of the Number_Of_States property is written, the size of the State_Text array is changed accordingly.

Test Concept: N1 and N2 are valid values of the Number_Of_States property, N1 and N2 do not equal the value of the Number_Of_States, and N1 does not equal N2. The value of the Number_Of_States property is written to N1 and the size of the State_Text and the value of the Number_Of_States property is verified. The procedure is repeated with N2 and again with N1.

Configuration Requirements: The IUT shall be configured with a Multi-state object O1, with a writable Number_Of_States property.

Test Steps:

1. WRITE O1, Number_Of_States = N1
2. VERIFY Number_Of_States = N1
3. VERIFY State_Text = N1, ARRAY INDEX = 0
4. WRITE O1, Number_Of_States = N2
5. VERIFY Number_Of_States = N2
6. VERIFY State_Text = N2, ARRAY INDEX = 0
7. WRITE O1, Number_Of_States = N1
8. VERIFY Number_Of_States = N1
9. VERIFY State_Text = N1, ARRAY INDEX = 0

7.3.2 Object Specific Tests

7.3.2.4 Averaging Object Tests

7.3.2.4.1 Reinitializing the Samples

Reason for Change: Add verify for initial values.

Purpose: To verify that an Averaging object correctly resets the Attempted_Samples, Valid_Samples, Minimum_Value, Average_Value, and Maximum_Value when Attempted_Samples, Object_Property_Reference, Window_Interval, or Window_Samples are changed.

Test Concept: The IUT is configured with an Averaging object that is actively monitoring some property value. The sampling is reinitialized by writing to the Attempted_Samples, Window_Interval, Window_Samples and Object_Property_Reference in turn. After each reinitialization verify that new sampling has begun.

Configuration Requirements: The IUT shall be configured with an Averaging object that is actively monitoring some property value. The sampling interval shall be long enough to permit the TD to verify that the sample is properly reinitialized.

Test Steps:

1. WRITE Attempted_Samples = 0,
2. *VERIFY Attempted_Samples = 0,*
3. *VERIFY Minimum_Value = INF,*
4. *VERIFY Maximum_Value = -INF,*
5. *VERIFY Average_Value = NaN,*
6. *VERIFY Valid_Samples = 0*

© 2026 by BACnet International. All rights reserved.

~~2~~7. WAIT (at least two sample times),
~~3~~8. VERIFY Minimum_Value = (a value x: -INF < x < INF),
~~4~~9. VERIFY Average_Value = (a value ≠ NaN),
~~5~~10. VERIFY Maximum_Value = (a value x: Minimum_Value <= x < INF),
~~6~~11. VERIFY Attempted_Samples = (a value x >= 2),
~~7~~12. VERIFY Valid_Samples = (a value x >= 2),
~~8~~13. WRITE Window_Interval = (any new value that will result in an appropriate sample time),
14. VERIFY Attempted_Samples = 0,
15. VERIFY Minimum_Value = INF,
16. VERIFY Maximum_Value = -INF,
17. VERIFY Average_Value = NaN,
18. VERIFY Valid_Samples = 0
~~9~~19. WAIT (at least two sample times),
~~10~~20. VERIFY Minimum_Value = (a value x: -INF < x < INF),
~~11~~21. VERIFY Average_Value = (a value ≠ NaN),
~~12~~22. VERIFY Maximum_Value = (a value x: Minimum_Value <= x < INF),
~~13~~23. VERIFY Attempted_Samples = (a value x >= 2),
~~14~~24. VERIFY Valid_Samples = (a value x >= 2),
~~15~~25. WRITE Window_Samples = (any new value that will result in an appropriate sample time),
26. VERIFY Attempted_Samples = 0,
27. VERIFY Minimum_Value = INF,
28. VERIFY Maximum_Value = -INF,
29. VERIFY Average_Value = NaN,
30. VERIFY Valid_Samples = 0
31. WAIT (at least two sample times),
32. VERIFY Minimum_Value = (a value x: -INF < x < INF),
33. VERIFY Average_Value = (a value ≠ NaN),
34. VERIFY Maximum_Value = (a value x: Minimum_Value <= x < INF),
35. VERIFY Attempted_Samples = (a value x >= 2),
36. VERIFY Valid_Samples = (a value x >= 2),
37. WRITE Window_Samples = (any new value that will result in an appropriate sample time),
~~16~~38. IF (Object_Property_Reference is writable) THEN {
39. WAIT (at least two sample times),
40. VERIFY Minimum_Value = (a value x: -INF < x < INF),
41. VERIFY Average_Value = (a value ≠ NaN),
42. VERIFY Maximum_Value = (a value x: Minimum_Value <= x < INF),
43. VERIFY Attempted_Samples = (a value x >= 2),
44. VERIFY Valid_Samples = (a value x >= 2),
45. WRITE Object_Property_Reference = (any new value),
46. IF (Samples_are_taken_immediately) THEN
47. VERIFY Attempted_Samples = 1,
48. VERIFY Minimum_Value = Average_Value,,
49. VERIFY Maximum_Value = Average_Value,
50. VERIFY Valid_Samples = 1
ELSE
51. VERIFY Attempted_Samples = 0,
52. VERIFY Minimum_Value = INF,
53. VERIFY Maximum_Value = -INF,
54. VERIFY Average_Value = NaN,
55. VERIFY Valid_Samples = 0
}

7.3.2.6 Binary Output Object Tests

[Remove test 7.3.2.6.3]

© 2026 by BACnet International. All rights reserved.

7.3.2.9 Command Object Tests

7.3.2.9.1 All Writes Successful with Post Delay Test

Reason for Change: Remove reference to externally visible output.

Purpose: To verify that a Command object can successfully execute an action list that includes post delays.

Test Concept: The IUT is configured with an action list that includes manipulating a sequence of *property values* ~~externally visible outputs~~ with a time delay between each *one*~~output~~. The TD triggers *the*~~this~~ action list and *verifies post delays are applied.* ~~the tester observes the external changes. If the IUT does not support Post Delay, then this test shall be omitted. If the IUT does not support action list configuration for this Test Concept, then this test shall be omitted.~~

Configuration Requirements: The IUT shall be configured with a Command object *O1* having *Action property containing* an action list, X, that includes writing *to property P1, with a value of V1, Post_Delay PD1, and another write to property P1, with a value of V2, and Post_Delay PD2. PD1 may equal PD2 but V1 must not equal V2. At the start of the test, P1 has a value V0 that is different than V1.* ~~to a sequence of externally visible outputs. There shall be a post delay between writes to the externally visible outputs that is long enough for the tester to observe the delay.~~ *PD1 and PD2 must be configured with a value large enough to perform the VERIFY steps between actions. If the IUT cannot meet these configuration requirements, this test shall be skipped.*

Test Steps:

1. WRITE Present_Value = *the array index for* X
~~2. RECEIVE BACnet-SimpleACK-PDU~~
   -- *(for any externally visible actions, verify that there is a post delay)*
2. BEFORE PD1 {
3. VERIFY In_Process = TRUE
4. VERIFY P1 = V1
   }
5. WAIT (the remainder of PD1)
6. BEFORE PD2 {
7. VERIFY In_Process = TRUE
8. VERIFY P1 = V2
   }
9. WAIT (the remainder of PD2)
~~4. CHECK (for the externally visible actions and verify that there is a post delay)~~
~~5~~*10.* VERIFY In_Process = FALSE
~~6~~*11.* VERIFY All_Writes_Successful = TRUE

7.3.2.9.2 Quit on Failure Test

Reason for Change: Remove reference to externally visible output.

Purpose: To verify that a Command object can successfully execute Quit_On_Failure procedures.

Test Concept: *The TD writes to the Present_Value of a command object O1 in the IUT that triggers the execution of an action list with multiple write operations, the first of which will fail and Quit_On_Failure is TRUE. The TD makes another write to Present_Value, triggering execution of another action list with multiple write operations, the first of which will fail and Quit_On_Failure is FALSE. The actions that occur after the failed write operation are observed to verify that they do not occur when Quit_On_Failure is TRUE and occur when Quit_On_Failure is FALSE.*~~The IUT is configured with two action lists that include a sequence of externally visible outputs with a write somewhere in the sequence that will fail. The action lists are identical except that one has Quit_On_Failure set to TRUE and the other set to FALSE. The TD triggers both action lists. The external outputs are observed to verify that the failure procedures are properly implemented. If the IUT does not support action list configuration for this Test Concept, then this test shall be omitted.~~

© 2026 by BACnet International. All rights reserved.

Configuration Requirements: *The IUT shall be configured with a Command object having at least two action lists, X and Y, each of which is configured with at least two discernable property/value write operations such that X(P1, V1) <> X(P2,V2) and Y(P3,V3) <> Y(P4,V4). The property/value write operations, X(P1,V1) and Y(P3,V3) will fail. Quit_On_Failure for X has a value of TRUE, Quit_On_Failure for Y has a value of FALSE. At the start of the test, each target property is configured with a value different than what the action will attempt to write and Present_Value of O1 <> X.* ~~The IUT shall be configured with a Command object having at least two action lists, X and Y, that includes writing to a sequence of externally visible outputs. Somewhere in the sequence there shall be a write command that will fail that is followed by write commands that will succeed. Both action lists shall be identical except that list X shall have Quit_On_Failure set to TRUE and Y shall have Quit_On_Failure set to FALSE. Delays shall be configured into the Action, if necessary, sufficient to let the tester observe that In_Process equals TRUE while the action is in process. If no Command object can be so configured, then verification of In_Process being TRUE shall be skipped~~

Test Steps:

1. *READ IV1 = P1 (Initial value of P1)*
2. *READ IV2 = P2 (Initial value of P2)*
3. WRITE Present_Value = *the array index for* X
4. *WAIT* ***Internal Processing Fail Time***
5. *WHILE (In_Process = TRUE) {}*

~~2. VERIFY In_Process = TRUE~~

~~3. CHECK (for the externally visible actions and verify that they stop when the failure occurs)~~

-- (for any externally visible actions, verify that they stop when the failure occurs)

~~4. VERIFY In_Process = FALSE,~~

6. VERIFY All_Writes_Successful = FALSE
7. *VERIFY P1 = IV1*
8. *VERIFY P2 = IV2*
9. *READ IV3 = P3 (Initial value of P3)*
10. *READ IV4 = P4 (Initial value of P4)*
11. WRITE Present_Value = *the array index for* Y
12. *WAIT* ***Internal Processing Fail Time***

~~7. VERIFY In_Process = TRUE~~

~~8. CHECK (for the externally visible actions and verify that they continue to the end after the failure occurs)~~

-- (for any externally visible actions, verify that they continue to the end after the failure occurs)

13. *WHILE (In_Process = TRUE) {}*

~~9. VERIFY In_Process = FALSE,~~

14. VERIFY All_Writes_Successful = FALSE
15. *VERIFY P3 = IV3*
16. *VERIFY P4 = V4 -- V4 is applied because Quit_On_Failure is false.*

7.3.2.9.4 Empty Action List Test

Reason for Change: Remove reference to externally visible output.

Purpose: To verify that a Command object takes no action when Present_Value is written to with a non-zero value that corresponds to an empty action list.

Test Concept: The IUT is configured with at least one empty action list. The TD triggers the action list. The external outputs are observed to verify that no changes occurred. If the IUT does not support action list configuration for this Test Concept, then this test shall be omitted.

Configuration Requirements: The IUT shall be configured with a Command object that has an Action property with at least one empty action list.

Test Steps:

© 2026 by BACnet International. All rights reserved.

1. WRITE Present_Value = (an index corresponding to an empty action list)
2. ~~RECEIVE BACnet-SimpleACK-PDU~~
3. ~~2~~. VERIFY In_Process = FALSE
4. ~~3~~. VERIFY All_Writes_Successful = TRUE
5. ~~CHECK (if any of the actions of the Command object have externally visible results verify that no changes occurred)~~
   -- *(if any of the actions of the Command object have externally visible results verify that no changes occurred)*

7.3.2.9.5 Action 0 Test

Reason for Change: Remove reference to externally visible output.

Purpose: To verify that a Command object takes no action when Present_Value is written to with a value of 0.

Test Concept: The Present_Value of a Command object, configured with at least one non-empty action list, is written with a value of 0. All_Writes_Successful is verified to have a value of TRUE.

Configuration Requirements: The IUT shall be configured with a Command object that has an Action property with at least one non-empty action list.

Test Steps:

1. WRITE Present_Value = 0
2. ~~RECEIVE BACnet-SimpleACK-PDU~~
2. *WAIT (**Internal Processing Fail Time**)*
3. VERIFY In_Process = FALSE
4. VERIFY All_Writes_Successful = TRUE
5. ~~CHECK (if any of the actions of the Command object have externally visible results verify that no changes occurred)~~
   -- *(if any of the actions of the Command object have externally visible results verify that no changes occurred)*

7.3.2.9.7 Write While In_Process is TRUE Test

Reason for Change: Remove reference to externally visible output.

Purpose: To verify that an action list continues to completion if a second action list is commanded while In_Process is TRUE and that the second action list is not executed.

Test Concept: The IUT is configured with two action lists that include *manipulating a property value. The TD triggers first action list and immediately triggers the second action list which must return an error. The manipulated property value is checked to ensure only the first action list was successful.* ~~a sequence of externally visible outputs with post delays for each action. The TD triggers the first action list. The external outputs are observed in order to trigger the second action list during the post delay of the first list. The TD triggers the second action list. The external outputs are observed to verify that the second action list is not executed. If the IUT does not support Post Delay, then this test shall be omitted.~~ If the IUT does not support action list configuration for this Test Concept, then this test shall be omitted.

Configuration Requirements: The IUT shall be configured with a Command object *O1* having two distinct action lists, X and Y, that include writing to *property P1 with X writing value V1 and Y writing value V2*~~a sequence of externally visible outputs~~. *Action list X will a contain post delay, PD1, sufficient to allow the TD time to write action list Y before timing out.*~~There shall be a post delay between writes to the externally visible outputs that is long enough for the tester to observe the delay (This ensures In_Process remains TRUE long enough to command the second action list).~~

Test Steps:

1. WRITE Present_Value = *the array index for* X
2. ~~WRITE Present_Value = Y~~

© 2026 by BACnet International. All rights reserved.

2. BEFORE PD1
3. TRANSMIT WriteProperty-Request,
'Object Identifier' = O1,
'Property Identifier' = Present_Value,
'Property Value' = Y
~~3. IF (Protocol_Revision is present and Protocol_Revision > 10) THEN~~
4. RECEIVE BACnet-Error-PDU,
Error Class = OBJECT,
Error Code = BUSY
~~ELSE~~
~~(RECEIVE BACnet-Error-PDU,~~
~~Error Class = OBJECT,~~
~~Error Code = BUSY) |~~
~~(RECEIVE BACnet-Error-PDU~~
~~Error Class = SERVICES,~~
~~Error Code = SERVICE_REQUEST_DENIED | OTHER)~~
~~4. CHECK (that the externally visible actions of X take place)~~
~~5. CHECK (that the externally visible actions of Y do not take place)~~
5. WAIT (PD1)
6. VERIFY P1 = V1
~~6~~*7*. VERIFY In_Process = FALSE,
~~7~~*8*. VERIFY All_Writes_Successful = TRUE

7.3.2.10 Device Object Tests

7.3.2.10.1 Active_COV_Subscriptions SubscribeCOV Test

Reason for change: Fixed the expected response for numeric and non-numeric.

Purpose: This test case verifies that the IUT correctly updates the Active_COV_Subscriptions property when COV subscriptions are created, cancelled and timed-out using SubscribeCOV.

~~Test Concept: INC1, INC2, and INC3 are each present if the property is numeric and optionally present otherwise.~~

Configuration Requirements: In this test, the tester shall choose three standard objects, O_1, O_2, and O_3, for which the device supports SubscribeCOV. O_1, O_2, and O_3 are not required to refer to different objects. The tester shall also choose three non-zero unique process identifiers, P_1, P_2, and P_3, and three non-zero lifetimes L_1, L_2, and L_3. Lifetime L_1 shall be long enough to allow the initial part of the test to run through to step *23*~~21~~. Lifetimes L_2 and L_3 shall be long enough for the whole test to be completed without expiring.

The IUT shall start the test with no entries in its Active_COV_Subscriptions property.

Test Steps:

1. TRANSMIT SubscribeCOV-Request,
'Subscriber Process Identifier' = P_1,
'Monitored Object Identifier' = O_1,
'Issue Confirmed Notifications' = TRUE,
'Lifetime' = L_1
2. RECEIVE BACnet-SimpleACK-PDU
3. BEFORE **Notification Fail Time**
4. RECEIVE ConfirmedCOVNotification-Request,
'Subscriber Process Identifier' = P_1,
'Initiating Device Identifier' = IUT,
'Monitored Object Identifier' = O_1,
'Time Remaining' = (a value approximately equal to L_1),

© 2026 by BACnet International. All rights reserved.

'List of Values' = (values appropriate to the object type of the monitored object)
5. TRANSMIT BACnet-SimpleACK-PDU
6. IF O_1, *Present_Value* ~~P_1~~ is numeric THEN
7. VERIFY Active_COV_Subscriptions = {S_1{ {TD, P_1}, {O_1, Present_Value }, TRUE, (a value less than L_1), (INC_1 : ~~not present or~~ a valid Increment) }}
ELSE
8. VERIFY Active_COV_Subscriptions = { S_1{ {TD, P1}, { O_1, Present_Value }, TRUE, (a value less than L_1), (INC_1: not present *or any Real value*)}}
9. TRANSMIT SubscribeCOV-Request,
'Subscriber Process Identifier' = P_2,
'Monitored Object Identifier' = O_2,
'Issue Confirmed Notifications' = FALSE,
'Lifetime' = L_2
10. RECEIVE BACnet-SimpleACK-PDU
11. BEFORE **Notification Fail Time**
12. RECEIVE UnconfirmedCOVNotification-Request,
'Subscriber Process Identifier' = P_2,
'Initiating Device Identifier' = IUT,
'Monitored Object Identifier' = O_2,
'Time Remaining' = (a value approximately equal to L_2),
'List of Values' = (values appropriate to the object type of the monitored object)
13. IF O_2, Present_Value is numeric THEN
~~13~~*14*. VERIFY Active_COV_Subscriptions = { S_1~~{ {TD, P_1}, {O_1, Present_Value}, TRUE, (a value less than L_1), INC_1~~},
S_2~~{~~ {TD, P_2}, {O_2, Present_Value}, FALSE, (a value less than L_2), (INC_2: ~~not present if the property is not numeric; present if~~ a valid Increment ~~was provided in the subscription; optionally present otherwise~~) } }
ELSE
15. VERIFY Active_COV_Subscriptions = {S_1, S_2{{TD, P_2}, {O_2, Present_Value}, FALSE, (a value less than L_2), (INC_2: not present or any Real value)}}
~~14~~*16*. TRANSMIT SubscribeCOV-Request,
'Subscriber Process Identifier' = P_3,
'Monitored Object Identifier' = O_3,
'Issue Confirmed Notifications' = FALSE,
'Lifetime' = L_3
~~15~~*17*. RECEIVE BACnet-SimpleACK-PDU
~~16~~*18*. BEFORE **Notification Fail Time**
~~17~~*19*. RECEIVE UnconfirmedCOVNotification-Request,
'Subscriber Process Identifier' = P_3,
'Initiating Device Identifier' = IUT,
'Monitored Object Identifier' = O_3,
'Time Remaining' = (a value approximately equal to L_3),
'List of Values' = (values appropriate to the object type of the monitored object)
~~18~~*20*. IF O_3, *Present_Value* ~~P_3~~ is numeric THEN
~~19~~*21*. VERIFY Active_COV_Subscriptions = { S_1~~{{TD, P_1}, {O_1, Present_Value}, TRUE, (a value less than L_1), INC_1~~},
S_2~~{{TD, P_2}, {O_2, Present_Value}, FALSE, (a value less than L_2), INC_2~~},
S_3{{TD, P_3}, {O_3, Present_Value}, FALSE, (a value less than L_3),
INC_3: : ~~not present or~~ a valid Increment}}
ELSE
~~20~~*22*. VERIFY Active_COV_Subscriptions = { S_1~~{{TD, P_1}, {O_1, Present_Value}, TRUE, (a value less than L_1), INC_1 }~~,
S_2~~{{TD, P_2,}, {O_2, Present_Value}, FALSE, (a value less than L_2),~~

© 2026 by BACnet International. All rights reserved.

~~INC$_2$ }~~,
S_3{{TD, P_3}, {O_3, Present_Value}, FALSE, (a value less than L_3),
(INC$_3$: not present *or any Real value*)}}
~~21~~*23*. WAIT L_1 + the IUT's timer granularity
~~22~~*24*. VERIFY Active_COV_Subscriptions = { S_2, S_3~~{{TD, P_2 }, {O_2 , Present_Value}, FALSE, (a value less than L_2),~~
~~INC$_2$ (a valid Increment if the property is REAL)},~~
~~{{TD, P_3 }, {O_3 , Present_Value}, FALSE, (a value less than L_3),~~
~~INC$_3$ (a valid Increment if the property is REAL))}~~}
~~23~~*25*. TRANSMIT SubscribeCOV-Request,
'Subscriber Process Identifier' = P_3,
'Monitored Object Identifier' = O_3
~~24~~*26*. RECEIVE BACnet-SimpleACK-PDU
~~25~~*27*. VERIFY Active_COV_Subscriptions = { S_2~~{{TD, P_2 }, {O_2 , Present_Value}, FALSE, (a value less than L_2),~~
~~INC$_2$ (a valid Increment if the property is REAL))}~~}
~~26~~*28*. TRANSMIT SubscribeCOV-Request,
'Subscriber Process Identifier' = P_2,
'Monitored Object Identifier' = O_2
~~27~~*29*. RECEIVE BACnet-SimpleACK-PDU
~~28~~*30*. VERIFY Active_COV_Subscriptions = { }

7.3.2.10.2 Active_COV_Subscriptions SubscribeCOVProperty Test

Reason for change: Moved test steps into the test instead of referencing 7.3.2.10.1.

Purpose: This test case verifies that the IUT correctly updates the Active_COV_Subscriptions property when COV subscriptions are created, cancelled and timed-out using SubscribeCOVProperty.

Configuration Requirements: In this test, the tester shall choose three objects and properties, O_1, O_2, and O_3, for which the device supports SubscribeCOVProperty. O_1, O_2, and O_3 are not required to refer to different objects. The tester shall also choose three non-zero unique process identifiers, P_1, P_2, and P_3, and three non-zero lifetimes, L_1, L_2 and L_3. Lifetime L_1 shall be long enough to allow the initial part of the test to run through to step *23*~~14~~. Lifetimes L_2 and L_3 shall be long enough for the whole test to be completed without expiring.

~~Test Steps: The test steps for this test case are identical to the test steps in Clause 7.3.2.10.1 except that SubscribeCOVProperty is used instead of SubscribeCOV and the Active_COV_Subscriptions entries shall reflect the property actually subscribed to (and not Present_Value if the subscribed property is not Present_Value).~~

The IUT shall start the test with no entries in its Active_COV_Subscriptions property.

Test Steps:

1. *TRANSMIT SubscribeCOVProperty-Request,*
   'Subscriber Process Identifier' = P_1,
   'Monitored Object Identifier' = O_1,
   'Issue Confirmed Notifications' = TRUE,
   'Lifetime' = L_1
   'Monitored Property Identifier' = MP_1,
   'COV Increment' = (any valid value -- optional)
2. *RECEIVE BACnet-SimpleACK-PDU*
3. *BEFORE **Notification Fail Time***
4. *RECEIVE ConfirmedCOVNotification-Request,*
   'Subscriber Process Identifier' = P_1,
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = O_1,
   'Time Remaining' = (a value approximately equal to L_1),
   'List of Values' = (value of MP_1 and the value of Status_Flags if present in O_1)

© 2026 by BACnet International. All rights reserved.

5. *TRANSMIT BACnet-SimpleACK-PDU*
6. *IF P_1 is numeric THEN*
7. *VERIFY Active_COV_Subscriptions = {S_1{{TD, P_1}, {O_1, MP_1}, TRUE, (a value less than L_1), (INC_1: a valid Increment)}}*

ELSE

8. *VERIFY Active_COV_Subscriptions = {S_1{{TD, P_1}, {O_1, MP_1}, TRUE, (a value less than L_1), (INC_1: not present or any Real value)}}*
9. *TRANSMIT SubscribeCOVProperty -Request,*
   'Subscriber Process Identifier' = P_2,
   'Monitored Object Identifier' = O_2,
   'Issue Confirmed Notifications' = FALSE,
   'Lifetime' = L_2
   'Monitored Property Identifier' = MP_2,
   'COV Increment' = (any valid value -- optional)
10. *RECEIVE BACnet-SimpleACK-PDU*
11. *BEFORE **Notification Fail Time***
12. *RECEIVE UnconfirmedCOVNotification-Request,*
    'Subscriber Process Identifier' = P_2,
    'Initiating Device Identifier' = IUT,
    'Monitored Object Identifier' = O_2,
    'Time Remaining' = (a value approximately equal to L_2),
    'List of Values' = (value of MP_2 and the value of Status_Flags if present in O_2)
13. *IF P_2 is numeric THEN*
14. *VERIFY Active_COV_Subscriptions = {S_1, S_2{{TD, P_2}, {O_2, MP_2}, FALSE, (a value less than L_2), (INC_2: a valid Increment)}}*

ELSE

15. *VERIFY Active_COV_Subscriptions = {S_1, S_2{{TD, P_2}, {O_2, MP_2}, FALSE, (a value less than L_2), (INC_2: not present or any Real value)}}*
16. *TRANSMIT SubscribeCOVProperty-Request,*
    'Subscriber Process Identifier' = P_3,
    'Monitored Object Identifier' = O_3,
    'Issue Confirmed Notifications' = FALSE,
    'Lifetime' = L_3
    'Monitored Property Identifier' = MP_3,
    'COV Increment' = (any valid value -- optional)
17. *RECEIVE BACnet-SimpleACK-PDU*
18. *BEFORE **Notification Fail Time***
19. *RECEIVE UnconfirmedCOVNotification-Request,*
    'Subscriber Process Identifier' = P_3,
    'Initiating Device Identifier' = IUT,
    'Monitored Object Identifier' = O_3,
    'Time Remaining' = (a value approximately equal to L_3),
    'List of Values' = (value of MP_3 and the value of Status_Flags if present in O_3)
20. *IF P_3 is numeric THEN*
21. *VERIFY Active_COV_Subscriptions = {S_1, S_2, S_3{{TD, P_3}, {O_3, MP_3}, FALSE, (a value less than L_3), INC_3: a valid Increment}}*

ELSE

22. *VERIFY Active_COV_Subscriptions = {S_1, S_2, S_3{{TD, P_3}, {O_3, MP_3}, FALSE, (a value less than L_3), (INC_3: not present or any Real value)}}*
23. *WAIT L_1 + the IUT's timer granularity*
24. *VERIFY Active_COV_Subscriptions = {S_2, S_3}*
25. *TRANSMIT SubscribeCOVProperty-Request,*

© 2026 by BACnet International. All rights reserved.

'Subscriber Process Identifier' = P_3,
'Monitored Object Identifier' = O_3
'Monitored Property Identifier' = MP_3,
'COV Increment' = (any valid value -- optional)

26. *RECEIVE BACnet-SimpleACK-PDU*
27. *VERIFY Active_COV_Subscriptions = {S_2}*
28. *TRANSMIT SubscribeCOVProperty-Request,*
'Subscriber Process Identifier' = P_2,
'Monitored Object Identifier' = O_2
'Monitored Property Identifier' = MP_2,
'COV Increment' = (any valid value -- optional)
29. *RECEIVE BACnet-SimpleACK-PDU*
30. *VERIFY Active_COV_Subscriptions = { }*

7.3.2.21 Notification Class Object

7.3.2.21.3 Recipient_List Tests

7.3.2.21.3.1 ~~ValidDays~~*Valid Days* Test

Reason for Change: Add a new Notes to Tester to account for possibility that the event-generating object's event algorithm does not support the pTimeDelay or pTimeDelayNormal parameters. Add a variable, O1, to represent the event generating object linked to the Notification Class object. Update the test to factor in all event algorithms and update to the new number schema. Move applicable conditionalities in Test Concept to a separate Test Conditionality and reword for clarity.

Reason For Change: Fix delay reference.

Purpose: To verify the operation of the Valid Days parameter of a BACnetDestination as used in the Recipient_List property of the Notification Class object.

Test Concept: The TD will select one instance of the Notification Class object and one instance of an event-generating ~~object~~*object, O1,* that is linked to the Notification Class object. The Recipient_List of the Notification Class object shall contain a single recipient with the Valid Days parameter configured so that at least one day is TRUE and at least one day is FALSE. The properties of ~~the event-generating object~~*O1* will be manipulated to cause the Event_State to ~~change from NORMAL to OFFNORMAL.~~*transition*. The tester verifies that if the local date is one of the valid days a notification message is transmitted and if the local date is not a valid day then no notification message is transmitted. ~~For devices that implement a read-only Recipient_List property for all instances of Notification Class objects and are exclusively configured for all days (Valid Days set to all Days), this test shall be omitted. For devices that implement a writeable Recipient_List property for all instances of Notification Class objects, and exclusively accept all days as the only permitted configuration, this test shall be omitted.~~

Test Conditionality: If the Valid Days parameter of the Recipient_List property of the Notification Class object only accepts all days as the only permitted configuration, this test shall be omitted.

Configuration Requirements: The IUT shall be configured with one or more instance of the Notification Class object and at least one event-generating ~~object~~*object, O1,* that is linked to the Notification Class object. ~~The event-generating object~~*O1* may be any object that supports intrinsic reporting or it may be an Event Enrollment object. ~~The event-generating object~~*O1* shall have the Event_Enable property configured to transmit notification messages for all event transitions. ~~The event-generating object~~*O1* shall be configured to be in a NORMAL event state at the start of the test. The Notification Class object shall be configured with a single recipient in the Recipient_List. The Valid Days parameter shall be configured so that at least one day of the week has a value of TRUE and at least one day of the week has a value of FALSE. The Transitions parameter shall be configured for the recipient to receive notifications for all event transitions. *D1 is either the pTimeDelay, or pTimeDelayNormal parameter, or 0) depending on the event transition.*

© 2026 by BACnet International. All rights reserved.

Notes to Tester: If O1's event algorithm does not support pTimeDelay or pTimeDelayNormal, assume pTimeDelay or pTimeDelayNormal equal zero when executing this test.

Test Steps:

1. (TRANSMIT TimeSynchronization-Request,
   'Time' = (any time within the window defined by From Time and To Time in the BACnetDestination that corresponds to one of the valid days)) |
   (TRANSMIT UTCTimeSynchronization-Request,
   'Time' = (any time within the window defined by From Time and To Time in the BACnetDestination that corresponds to one of the valid days, converted to UTC)) |
   MAKE (the local date and time = (any time within the window defined by From Time and To Time in the BACnetDestination that corresponds to one of the valid days))
2. ~~WAIT (pTimeDelay + **Notification Fail Time**)~~*WAIT (**Internal Processing Fail Time**)*
3. VERIFY pCurrentState = NORMAL
4. IF (pMonitoredValue is writable) THEN
5. WRITE pMonitoredValue = (~~a value that is OFFNORMAL~~*a value which will result in the transition of O1*)
   ELSE
6. MAKE (pMonitoredValue have ~~a value that is OFFNORMAL~~*a value which will result in the transition of O1*)
~~5.~~*7.* WAIT (~~pTimeDelay~~*D1*)
~~6.~~*8.* BEFORE **Notification Fail Time**
9. RECEIVE ConfirmedEventNotification-Request,
   'Process Identifier' = (any valid process ID),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (~~the event-generating object configured for this test~~*O1*),
   'Time Stamp' = (any valid time stamp),
   'Notification Class' = (~~the class corresponding to the object being tested~~*the notification class configured for O1*),
   'Priority' = (~~the value configured to correspond to a TO-OFFNORMAL transition~~*the value configured for the transition*),
   'Event Type' = (any valid event type),
   'Message Text' = (optional, any valid message text),
   'Notify Type' = EVENT | ALARM,
   'AckRequired' = TRUE | FALSE,
   'From State' = NORMAL,
   'To State' = ~~OFFNORMAL,~~*(ES1: any event state appropriate to the event type),*
   'Event Values' = (values appropriate to the event type)
~~7.~~*10.* TRANSMIT BACnet-SimpleACK-PDU
~~8.~~*11.* VERIFY pCurrentState = ~~OFFNORMAL~~*ES1*
~~9.~~*12.* (TRANSMIT TimeSynchronization-Request,
   'Time' = (any time within the window defined by From Time and To time in the BACnetDestination that corresponds to one of the invalid days)) |
   (TRANSMIT UTCTimeSynchronization-Request,
   Time' = (any time within the window defined by From Time and To Time in the BACnetDestination that corresponds to one of the invalid days, converted to UTC)) |
   MAKE (the local date and time = (any time within the window defined by From Time and To Time in the BACnetDestination that corresponds to one of the invalid days))
~~10.~~*13.* IF (pMonitoredValue is writable) THEN
14. WRITE pMonitoredValue = (~~a value that is NORMAL~~*a value which will result in the transition of O1*)
   ELSE
15. MAKE (pMonitoredValue have ~~a value that is NORMAL~~*a value which will result in the transition of O1*)
~~11.~~*16.* WAIT (~~pTimeDelay~~*D1* + **Notification Fail Time**)
~~12.~~*17.* CHECK (verify that no notification message was transmitted)

© 2026 by BACnet International. All rights reserved.

7.3.2.21.3.2 ~~FromTime and ToTime Test~~*From Time and To Time Test*

Reason for Change: Add a new Notes to Tester to account for possibility that the event-generating object's event algorithm does not support the pTimeDelay or pTimeDelayNormal parameters. Update test to use new numbering scheme. Move specified conditionalities in Test Concept to be in a separate Test Conditionality and reword for clarity.

Purpose: To verify the operation of the From Time and To Time parameters of a BACnetDestination as used in the Recipient_List property of the Notification Class object.

Test Concept: The case where the local date and time fall within the window defined by the From Time and To Time parameters is covered by the ~~ValidDays~~*Valid Days* test in 7.3.2.21.3.1. This test uses the same IUT configuration and sets the local time to a value that is one of the ~~ValidDays~~*Valid Days* but outside of the window defined by the From Time and To Time parameters. The objective is to verify that an event notification message is not transmitted when the event is triggered. ~~For devices that implement a read-only Recipient_List property for all instances of Notification Class objects and are exclusively configured for all times (From Time set to 00:00:00.0, To Time set to 23:59:59.99), this test shall be omitted. For devices that implement a writeable Notification Class Recipient_List property for all instances of Notification Class objects, and exclusively accept all times as the only permitted configuration, this test shall be omitted.~~

Test Conditionality: If the From Time and To Time parameters of the Recipient_List property of the Notification Class object only support 0:00:00.0 and 23:59:59.99 respectively, this test shall be omitted.

~~Configuration Requirements: The configuration requirements are identical to the requirements in 7.3.2.21.3.1.~~
Configuration Requirements: The IUT shall be configured with one or more instance of the Notification Class object and at least one event-generating object, O1, that is linked to the Notification Class object. O1 may be any object that supports intrinsic reporting or it may be an Event Enrollment object. O1 shall have the Event_Enable property configured to transmit notification messages for all event transitions. O1 shall be configured to be in a NORMAL event state at the start of the test. The Notification Class object shall be configured with a single recipient in the Recipient_List. The IUT is configured at a time that is one of the Valid Days, but outside the window defined by the From Time and To Time parameters. The Transitions parameter shall be configured for the recipient to receive notifications for all event transitions. D1 is either the pTimeDelay, or pTimeDelayNormal parameter, or 0 depending on the event transition.

Notes to Tester: If the event generating object's event algorithm does not support pTimeDelay or pTimeDelayNormal, assume pTimeDelay or pTimeDelayNormal equal zero when executing this test.

Test Steps:

1. (TRANSMIT TimeSynchronization-Request,
   'Time' = (any time outside the window defined by From Time and To Time in the BACnetDestination that corresponds to one of the valid days)) |
   (TRANSMIT UTCTimeSynchronization-Request,
   'Time' = (any time outside the window defined by From Time and To Time in the BACnetDestination that corresponds to one of the valid days, converted to UTC)) |
   MAKE (the local date and time = (any time outside the window defined by From Time and To Time in the BACnetDestination that corresponds to one of the valid days))
2. WAIT (~~pTimeDelay~~*D1* + Notification Fail Time)
3. VERIFY pCurrentState = NORMAL
4. IF (pMonitoredValue is writable) THEN
5. WRITE pMonitoredValue = (~~a value that is OFFNORMAL~~*a value which will result in the transition of the event generating object*)
   ELSE
6. MAKE (pMonitoredValue have ~~a value that is OFFNORMAL~~*a value which will result in the transition of the event generating object)*
7. WAIT (~~pTimeDelay~~*D1* + Notification Fail Time)
8. CHECK (verify that no notification message was transmitted)

[Remove test 7.3.2.21.3.4]

© 2026 by BACnet International. All rights reserved.

7.3.2.21.3.5 Recipient_List Property Supports Device Identifier Recipients Test

Reason for Change: Add a new Notes to Tester to account for possibility that the event-generating object's event algorithm does not support the pTimeDelay or pTimeDelayNormal parameters. Move the existing Notes to Tester from after the Test Steps to before the Test Steps. Update to the new numbering schema. Add missing space after Configuration Requirements colon. Remove test conditionality regarding a read only Recipient_List property because this test is executed when Supports Writable Recipient_List is checked.

Purpose: To verify that the Recipient_List property of the Notification Class object supports the device form of the Recipient component and that the IUT is able to associate a MAC address with the Device Identifier. The intent is to ensure that the IUT is able to locate the specified alarm recipient and send notification to the specified recipient. ~~This test shall be run if the IUT's Notification Class object's Recipient_List property supports the BACnet object identifier form of BACnetRecipient.~~

Test Concept: The tester shall select a single event generating object E in the IUT that references Notification Class object N. The tester shall add an entry into the Recipient_List of the associated Notification Class object that specifies a Device Identifier, D, for a device that the IUT is not already aware of. The TD, acting as device D, shall be located on a different network than the IUT to ensure that the IUT is capable of binding to recipients located on any network. ~~For devices that implement a read-only Recipient_List property for all instances of Notification Class objects, and there is an address form of the Recipient component in all instances, this test shall be omitted.~~ D1 is either the pTimeDelay, or pTimeDelayNormal parameter, or 0 (for transitions to and from FAULT state) depending on the event transition.

Configuration ~~Requirements:The~~*Requirements: The* TD shall be configured so that it does not execute WhoHas.

Notes to Tester: If the event generating object's event algorithm does not support pTimeDelay or pTimeDelayNormal, assume pTimeDelay or pTimeDelayNormal equal zero when executing this test.

Notes to Tester: The IUT is expected to initiate one or more range-restricted WhoIs requests after the modification of the Recipient_List but before the sending of the notification. The IUT might also need to perform other network discovery operations. Given that there are multiple approaches to the use of WhoIs for device discovery, the test only focuses on the IUT's ability to find device D and not on the specifics or timing of the WhoIs requests.

Test Steps:

1. WRITE N.RecipientList = ({all days, all times, D, any process ID, FALSE, all transitions})
2. MAKE (a condition exist that will cause E to generate an event transition)
3. WAIT D1
4. BEFORE (Notification Fail Time plus the amount of time the IUT takes to perform device discovery)
5. RECEIVE UnconfirmedEventNotification-Request,
   'Process Identifier' = ~~PID~~*any valid process ID*,
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = E,
   'Time Stamp' = (any valid time stamp),
   'Notification Class' = (N's instance),
   'Priority' = (any valid priority),
   'Event Type' = (any valid event type),
   'Message Text' = (optional, any valid message text),
   'Notify Type' = ALARM | EVENT,
   'AckRequired' = TRUE | FALSE,
   'From State' = (any valid event state),
   'To State' = (any valid event state),
   'Event Values' = (values appropriate to the event type)

~~Notes to Tester: The IUT is expected to initiate one or more range-restricted WhoIs requests after the modification of the Recipient_List but before the sending of the notification. The IUT might also need to perform other network discovery~~

© 2026 by BACnet International. All rights reserved.

~~operations. Given that there are multiple approaches to the use of WhoIs for device discovery, the test only focuses on the IUT's ability to find device D and not on the specifics or timing of the WhoIs requests.~~

7.3.2.21.3.6 Recipient_List Property Supports Network Address Recipients

Reason for Change: Add a new Notes to Tester to account for possibility that the event-generating object's event algorithm does not support the pTimeDelay or pTimeDelayNormal parameters. Update test to new numbering schema. Remove test conditionality regarding a read only Recipient_List property because this test is executed when Supports Writable Recipient_List is checked.

Purpose: To verify that the Recipient_List property of the Notification Class object supports the address form of the Recipient component. The intent is to ensure that the IUT is able to send notifications to the specified recipient.

Test Concept: The tester shall select a single event-generating object E in the IUT that references Notification Class object N. The tester shall add an entry into the Recipient_List of the associated Notification Class object that specifies a BACnetAddress A, where A is a unicast or is a local, remote, or global broadcast address. ~~For devices that implement a readonly Recipient_List property for all instances of Notification Class objects, and there is a Device Identifier form of the Recipient component in all instances, this test shall be skipped.~~ D1 is either the pTimeDelay, or pTimeDelayNormal parameter, or 0 (for transitions to and from FAULT state) depending on the event transition.

Notes to Tester: If the event generating object's event algorithm does not support pTimeDelay or pTimeDelayNormal, assume pTimeDelay or pTimeDelayNormal equal zero when executing this test.

Test Steps:
1. WRITE N.RecipientList = ({all days, all times, A, any process ID, FALSE, all transitions})
2. MAKE (a condition exist that will cause E to generate an event transition)
3. WAIT D1
4. BEFORE Notification Fail Time
5. RECEIVE
   DESTINATION = A,
   UnconfirmedEventNotification-Request,
   'Process Identifier' = ~~PID~~*any valid process ID*,
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = E,
   'Time Stamp' = (any valid time stamp),
   'Notification Class' = (N's instance),
   'Priority' = (any valid priority),
   'Event Type' = (any valid event type),
   'Message Text' = (optional, any valid message text),
   'Notify Type' = ALARM | EVENT,
   'AckRequired' = TRUE | FALSE,
   'From State' = (any valid event state),
   'To State' = (any valid event state),
   'Event Values' = (values appropriate to the event type)

7.3.2.21.3.X1 Transitions Test Normal-to-Offnormal

Reason for Change: No test exists for this functionality.

Purpose: To verify that notification messages are transmitted only if the bit in the Transitions parameter corresponding to the event transition is set for TO_OFFNORMAL or TO_NORMAL event transitions.

© 2026 by BACnet International. All rights reserved.

Test Concept: The IUT shall be configured such that TO_OFFNORMAL event transitions generate event notifications and TO_NORMAL event transitions do not generate notifications. TO_OFFNORMAL and TO_NORMAL event transitions shall be triggered, and the IUT shall be monitored to verify that notification messages are transmitted only for TO_OFFNORMAL transitions. The test shall then be repeated with the TO_OFFNORMAL and TO_NORMAL Transition settings reversed.

Test Conditionality: This test shall be skipped if the IUT does not contain a normal to offnormal event generating object or the IUT exclusively accepts all transitions as the only permitted configuration for the Transitions parameter of the Notification Class Recipient_List property.

Configuration Requirements: The IUT shall be configured with one or more instance of the Notification Class object and at least one event-generating object that is linked to the Notification Class object under test, NC1. The event-generating object, O1, may be any normal to offnormal event generating object. O1 shall have the Event_Enable property configured to transmit messages for all event transitions. O1 shall be configured to be in a NORMAL event state at the start of the test. NC1 shall be configured with a single recipient, R, in the Recipient_List property with the Transitions parameter set to (TRUE, FALSE, FALSE). The local date and time shall be configured such that the current day is included in VD, where VD is the ValidDays value, and the current time falls within TW, where TW is the time window defined by the From Time and To Time values.

Notes to Tester: For algorithms that do not apply the pTimeDelay or pTimeDelayNormal parameters, assume pTimeDelay or pTimeDelayNormal equals zero when executing this test.

Notes to Tester: The UnconfirmedEventNotification service may be substituted for the ConfirmedEventNotification service. When executing the test using the UnconfirmedEventNotification service, the value written for the Recipient_List's Issue Confirmed Notification parameter shall be FALSE instead of TRUE. The TD shall also skip all steps requiring transmission of a BACnet-SimpleACK-PDU.

Test Steps:

1. VERIFY pCurrentState = NORMAL
2. WAIT (pTimeDelay + Notification Fail Time)
3. IF (pMonitoredValue is writable) THEN
4. WRITE pMonitoredValue = (a value that is OFFNORMAL)
   ELSE
5. MAKE (pMonitoredValue have a value that is OFFNORMAL)
6. WAIT (pTimeDelay)
7. BEFORE Notification Fail Time
8. RECEIVE ConfirmedEventNotification-Request
   'Process Identifier' = (any valid process ID)
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (O1)
   'Time Stamp' = (any valid time stamp),
   'Notification Class' = (NC1)
   'Priority' = (the value configured to correspond to a TO_OFFNORMAL transition),
   'Event Type' = (any valid event type),
   'Notify Type' = EVENT | ALARM,
   'AckRequired' = TRUE | FALSE,
   'From State' = NORMAL,
   'To State' = OFFNORMAL,
   'Event Values' = (values appropriate to the event type)
9. TRANSMIT BACnet-SimpleACK-PDU
10. VERIFY pCurrentState = OFFNORMAL
11. IF (pMonitoredValue is writable) THEN
12. WRITE pMonitoredValue = (a value that is NORMAL)
    ELSE
13. MAKE (pMonitoredValue have a value that is NORMAL)
14. WAIT (pTimeDelayNormal + Notification Fail Time)
15. CHECK (verify that the IUT did not transmit an event notification message)

© 2026 by BACnet International. All rights reserved.

16. VERIFY pCurrentState = NORMAL
17. WRITE NC1.RecipientList = ({VD, TW, R, PID1, TRUE, (FALSE, FALSE, TRUE)})
18. IF (pMonitoredValue is writable) THEN
19. WRITE pMonitoredValue = (a value that is OFFNORMAL)
    ELSE
20. MAKE (pMonitoredValue have a value that is OFFNORMAL)
21. WAIT (pTimeDelay + Notification Fail Time)
22. CHECK (verify that the IUT did not transmit an event notification message)
23. VERIFY pCurrentState = OFFNORMAL
24. IF (pMonitoredValue is writable) THEN
25. WRITE pMonitoredValue = (a value that is NORMAL)
    ELSE
26. MAKE (pMonitoredValue have a value that is NORMAL)
27. WAIT (pTimeDelayNormal)
28. BEFORE Notification Fail Time
29. RECEIVE ConfirmedEventNotification-Request
    'Process Identifier' = (PID1)
    'Initiating Device Identifier' = IUT,
    'Event Object Identifier' = (O1)
    'Time Stamp' = (any valid time stamp),
    'Notification Class' = (NC1)
    'Priority' = (the value configured to correspond to a TO_NORMAL transition),
    'Event Type' = (any valid event type),
    'Notify Type' = EVENT | ALARM,
    'AckRequired' = TRUE | FALSE,
    'From State' = OFFNORMAL,
    'To State' = NORMAL,
    'Event Values' = (values appropriate to the event type)
30. TRANSMIT BACnet-SimpleACK-PDU
31. VERIFY pCurrentState = NORMAL

7.3.2.21.3.X2 Transitions Test Normal-to-Fault

Reason for Change: No test exists for this functionality.

Purpose: To verify that notification messages are transmitted only if the bit in the Transition parameter corresponding to the event transitions is set for TO_FAULT or TO_NORMAL event transitions.

Test Concept: The IUT shall be configured such that TO_FAULT event transitions generate event notifications and TO_NORMAL event transitions do not generate notifications. TO_FAULT and TO_NORMAL event transitions shall be triggered, and the IUT shall be monitored to verify that notification messages are transmitted only for TO_FAULT transitions. The test shall then be repeated with the TO_FAULT and TO_NORMAL Transition settings reversed.

Test Conditionality: This test shall be skipped if the IUT does not contain a normal to fault event generating object or the IUT exclusively accepts all transitions as the only permitted configuration for the Transitions parameter of the Notification Class Recipient_List property.

Configuration Requirements: The IUT shall be configured with one or more instance of the Notification Class object and at least one event-generating object that is linked to the Notification Class object under test, NC1. The event-generating object, O1, may be any normal to fault event generating object. O1 shall have the Event_Enable property configured to transmit messages for all event transitions. O1 shall be configured to be in a NORMAL event state at the start of the test. NC1 shall be configured with a single recipient in the Recipient_List property with the Transitions parameter set to (FALSE, TRUE, FALSE). The local date and time shall be configured such that the current day matches a day specified by the ValidDays parameter and the current time is between the From Time and To Time parameters.

© 2026 by BACnet International. All rights reserved.

Notes to Tester: The UnconfirmedEventNotification service may be substituted for the ConfirmedEventNotification service. When executing the test using the UnconfirmedEventNotification service, the value written for the Recipient_List's Issue Confirmed Notification parameter shall be FALSE instead of TRUE. The TD shall also skip all steps requiring transmission of a BACnet-SimpleACK-PDU.

Test Steps:

1. VERIFY pCurrentState = NORMAL
2. WAIT (pTimeDelay + Notification Fail Time)
3. MAKE (a condition exist that will cause the object to generate a TO_FAULT transition)
4. BEFORE Notification Fail Time
5. RECEIVE ConfirmedEventNotification-Request
   'Process Identifier' = (any valid process ID)
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (O1)
   'Time Stamp' = (any valid time stamp),
   'Notification Class' = (NC1)
   'Priority' = (the value configured to correspond to a TO_FAULT transition),
   'Event Type' = (any valid event type),
   'Notify Type' = EVENT | ALARM,
   'AckRequired' = TRUE | FALSE,
   'From State' = NORMAL,
   'To State' = FAULT,
   'Event Values' = (values appropriate to the event type)
6. TRANSMIT BACnet-SimpleACK-PDU
7. VERIFY pCurrentState = FAULT
8. MAKE (a condition exist that will cause the object to generate a TO_NORMAL transition)
9. WAIT (pTimeDelayNormal + Notification Fail Time)
10. CHECK (verify that the IUT did not transmit an event notification message)
11. WRITE NC1.RecipientList = ({VD, TW, R, PID1, TRUE, (FALSE, FALSE, TRUE)})
12. MAKE (a condition exist that will cause the object to generate a TO_FAULT transition)
13. WAIT Notification Fail Time
14. CHECK (verify that the IUT did not transmit an event notification message)
15. MAKE (a condition exist that will cause the object to generate a TO_NORMAL transition)
16. WAIT (pTimeDelayNormal)
17. BEFORE Notification Fail Time
18. RECEIVE ConfirmedEventNotification-Request
   'Process Identifier' = (PID1)
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (O1)
   'Time Stamp' = (any valid time stamp),
   'Notification Class' = (NC1)
   'Priority' = (the value configured to correspond to a TO_NORMAL transition),
   'Event Type' = (any valid event type),
   'Notify Type' = EVENT | ALARM,
   'AckRequired' = TRUE | FALSE,
   'From State' = FAULT,
   'To State' = NORMAL,
   'Event Values' = (values appropriate to the event type)
19. TRANSMIT BACnet-SimpleACK-PDU
20. VERIFY pCurrentState = NORMAL

© 2026 by BACnet International. All rights reserved.

7.3.2.21.3.X3 Transitions Test Normal-to-Normal

Reason for Change: No test exists for this functionality.

Purpose: To verify that notification messages are transmitted only if the bit in the Transitions parameter corresponding to the event transition is set for TO_NORMAL event transitions.

Test Concept: The IUT shall be configured such that TO_NORMAL event transitions do not generate notifications. A TO_NORMAL event transitions shall be triggered, and the IUT shall be monitored to verify that notification messages are not transmitted. The Recipient_List's Transitions parameter shall then be written, such that TO_NORMAL event transitions trigger a notification message to be transmitted.

Test Conditionality: This test shall be skipped if the IUT does not contain a normal-to-normal event generating object or the IUT exclusively accepts all transitions as the only permitted configuration for the Transitions parameter of the Notification Class Recipient_List property.

Configuration Requirements: The IUT shall be configured with one or more instance of the Notification Class object and at least one event-generating object that is linked to the Notification Class object under test, NC1. The event-generating object, O1, may be any normal-to-normal event generating object. O1 shall have the Event_Enable property configured to transmit messages for all event transitions. O1 shall be configured to be in a NORMAL event state at the start of the test. NC1 shall be configured with a single recipient, R, in the Recipient_List property with the Transitions parameter set to (FALSE, FALSE, FALSE). The local date and time shall be configured such that the current day is included in VD, where VD is the ValidDays value, and the current time falls within TW, where TW is the time window defined by the From Time and To Time values.

Notes to Tester: For algorithms that do not apply the pTimeDelay or pTimeDelayNormal parameters, assume pTimeDelay or pTimeDelayNormal equals zero when executing this test.

Notes to Tester: The UnconfirmedEventNotification service may be substituted for the ConfirmedEventNotification service. When executing the test using the UnconfirmedEventNotification service, the value written for the Recipient_List's Issue Confirmed Notification parameter shall be FALSE instead of TRUE. The TD shall also skip all steps requiring transmission of a BACnet-SimpleACK-PDU.

Test Steps:

1. VERIFY pCurrentState = NORMAL
2. WAIT (pTimeDelay + Notification Fail Time)
3. IF (pMonitoredValue is writable) THEN
4. WRITE pMonitoredValue = (a value that will result in a TO_NORMAL transition)
   ELSE
5. MAKE (pMonitoredValue have a value that will result in a TO_NORMAL transition)
6. WAIT (pTimeDelayNormal + Notification Fail Time)
7. CHECK (verify that the IUT did not transmit an event notification message)
8. VERIFY pCurrentState = NORMAL
9. WRITE NC1.RecipientList = ({VD, TW, R, PID1, TRUE, (FALSE, FALSE, TRUE)})
10. IF (pMonitoredValue is writable) THEN
11. WRITE pMonitoredValue = (a value that will result in a TO_NORMAL transition)
    ELSE
12. MAKE (pMonitoredValue have a value that will result in a TO_NORMAL transition)
13. WAIT (pTimeDelayNormal)
14. BEFORE Notification Fail Time
15. RECEIVE ConfirmedEventNotification-Request
    'Process Identifier' = (PID1)
    'Initiating Device Identifier' = IUT,
    'Event Object Identifier' = (O1)
    'Time Stamp' = (any valid time stamp),

© 2026 by BACnet International. All rights reserved.

'Notification Class' = (NC1)
'Priority' = (the value configured to correspond to a TO_NORMAL transition),
'Event Type' = (any valid event type),
'Notify Type' = EVENT | ALARM,
'AckRequired' = TRUE | FALSE,
'From State' = NORMAL,
'To State' = NORMAL,
'Event Values' = (values appropriate to the event type)

16. TRANSMIT BACnet-SimpleACK-PDU
17. VERIFY pCurrentState = NORMAL

7.3.2.23 Schedule Object Tests

[Remove test 7.3.2.23.1]

[Remove test 7.3.2.23.2]

[Remove test 7.3.2.23.3.1]

[Remove test 7.3.2.23.3.2]

[Remove test 7.3.2.23.3.3]

[Remove test 7.3.2.23.3.4]

[Remove test 7.3.2.23.3.5]

[Remove test 7.3.2.23.3.6]

[Remove test 7.3.2.23.3.7]

[Remove test 7.3.2.23.3.8]

[Remove test 7.3.2.23.3.9]

[Remove test 7.3.2.23.4]

[Remove test 7.3.2.23.7]

[Remove test 7.3.2.23.8]

7.3.2.24 Logging Object Tests

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

© 2026 by BACnet International. All rights reserved.

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)

7.3.2.25 Event Log Tests

7.3.2.25.1 Internal Logging of Notifications

Reason for change: Fixed minor errors

Purpose: To verify the IUT correctly collects and represents the Notifications which it initiates.

Test Concept: Make the IUT generate two event notification messages which the IUT logs. Use ReadRange to retrieve them from an Event Log and compare the two representations.

Configuration Requirements: The tester shall choose two events which are configured to be sent to the TD and to be placed into one of the IUT's Event Logs, LO1.

Notes to Tester: When the UnconfirmedEventNotification service is used instead of the ConfirmedEventNotification service,the TD shall skip the steps in which a BACnet-SimpleACK-PDU is sent.

Test Steps:
1. WRITE Enable = TRUE
2. MAKE (a condition exist that will cause the device to generate an event transition)
3. WAIT ~~D1~~ ***Notification Fail Time***
4. *EV1* = RECEIVE ConfirmedEventNotification-Request,
   'Process Identifier' = (any valid process identifier),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (any valid object),
   'Time Stamp' = *(*T1, any valid timestamp),
   'Notification Class' = (any valid notification class),
   'Priority' = (any valid priority),
   'Event Type' = (any standard event type),
   'Message Text' = (optional, any valid message text),
   'Notify Type' = ALARM | EVENT,
   'AckRequired' = TRUE | FALSE,
   'From State' = (state S1, any valid state for this event type),
   'To State' = (state S2, any valid state for this event type that can follow S1),
   'Event Values' = (any values appropriate to the event type)
5. TRANSMIT BACnet-SimpleACK-PDU
6. MAKE (a condition exist that will cause the device to generate an event transition)
7. WAIT ~~D2~~ ***Notification Fail Time***
8. *EV2* = RECEIVE ConfirmedEventNotification-Request,
   'Process Identifier' = (any valid process identifier),
   'Initiating Device Identifier' = IUT,

© 2026 by BACnet International. All rights reserved.

'Event Object Identifier' = (any valid object),
'Time Stamp' = (T2, any valid timestamp),
'Notification Class' = (any valid notification class),
'Priority' = (any valid priority),
'Event Type' = (any standard event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = ALARM | EVENT,
'AckRequired' = TRUE | FALSE,
'From State' = (state S3, any valid state for this event type),
'To State' = (state S4, any valid state for this event type that can follow S3),
'Event Values' = (any values appropriate to the event type)

9. TRANSMIT BACnet-SimpleACK-PDU
10. READ RC = LO1, Record_Count
11. TRANSMIT ReadRange-Request,
    'Object Identifier' = LO1,
    'Property Identifier' = Log_Buffer,
    'Reference Index'= RC,
    'Count'= -2
12. RECEIVE ReadRange-ACK,
    'Object Identifier' = LO1,
    'Property Identifier' = Log_Buffer,
    'Result Flags' = {?, ?, FALSE},
    'Item Count' = 2,
    'Item Data' = (logged data that matches *EV1 and EV2,* ~~the information received in steps 3 and 6,~~ except that Process_Identifier may be any value and is not required to match)
13. CHECK (T2 > T1, and that the notifications were logged in order)

7.3.2.25.X1 Logging Events Never Transmitted as an APDU

Reason for change: No tests exist for this functionality.

Purpose: To verify the IUT correctly stores event log records for event notifications that are never transmitted as an APDU.

Test Concept: The IUT is made to generate an event that is logged in the event log but is not intended for transmission as an APDU, such that the associated notification-class object is uninitialized (does not exist in the IUT). The event log record is read to verify it is stored as a proper BACnetEventLogRecord.

Configuration Requirements: Event Log, LO1 is configured to log the event notification(s) generated in the test.

Test Steps:

1. READ RC = LO1, Record_Count
2. MAKE (the device generate an event that gets entered into the log buffer of LO1)
3. TRANSMIT ReadRange-Request,
   ‘Object Identifier’= LO1,
   ‘Property Identifier’= Log_Buffer,
   ‘Reference Index’= RC+1,
   ‘Count’= -1
4. RECEIVE ReadRange-ACK,
   ‘Object Identifier’= LO1,
   ‘Property Identifier’= Log_Buffer,
   ‘Result Flags’= {?,?,FALSE},
   ‘Item Count’= 1
   ‘Item Data’=
      ‘Timestamp’= (any valid timestamp),

© 2026 by BACnet International. All rights reserved.

‘Log-datum’= notification,
‘Process Identifier’= (any valid value)
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (any valid object),
'Time Stamp' = (any valid timestamp),
'Notification Class' = (a value that does not correspond to a notification-class instance in the IUT),
'Priority' = (any valid priority),
'Event Type' = (any standard event type),
'Message Text' = (optional, any valid message text),
'Notify Type' = ALARM | EVENT,
'AckRequired' = TRUE | FALSE,
'From State' = (any valid state for this event type),
'To State' = (any valid state for this event type),
'Event Values' = (any values appropriate to the event type)

7.3.2.30 Notification Forwarder Object Tests

7.3.2.30.8 Subscribed_Recipients Property Test

7.3.2.30.8.1 Time Count Down Test

Reason for change: Vendor specified time remaining was not clearly defined. CHECKs were used in the test instead of VERIFY.

Purpose: Verify Subscribed_Recipients entries count Time Remaining down.

Test Concept: Add a subscription to a Notification Forwarder and check that the Subscribed_Recipients Time Remaining value decreases after time has passed.

Note to tester: If the device timing resolution is ~~significantly~~ greater than 1 minute, ~~the~~ wait *the time specified by the vendor to cause the time remaining to decrease* ~~times called for by the test steps may be extended~~.

Configuration Requirements: Base setup 2 for Notification Forwarder object tests with TR lifetime sufficient for this test.

Test Steps:

1. READ SR1 = Subscribed_Recipients
~~2. CHECK (SR1 = {DEST_OBJ_ID, Recipient D1~~
~~DEST_PROCESS_ID, Process Identifier~~
~~FALSE, Issue Confirmed Notifications~~
~~TR1 Time Remaining where (TR-1) <= TR1 <= TR~~
~~}) One list element~~
~~3~~*2*. WAIT greater of 2 minutes or *the vendor specified time* ~~twice timing resolution~~
~~4~~*3*. READ SR2 = Subscribed_Recipients
4. VERIFY (SR2, time-remaining < SR1, time-remaining)
~~5. CHECK (SR2 = {DEST_OBJ_ID, Recipient D1~~
~~DEST_PROCESS_ID, Process Identifier~~
~~FALSE, Issue Confirmed Notifications~~
~~TR2 -- Time Remaining where TR2 < TR1~~
~~}) One list element~~

© 2026 by BACnet International. All rights reserved.

7.3.2.30.13 Persistence Tests

7.3.2.30.13.1 Recipient_List Persistence Test

Reason for Change: Moved closing curly brackent from after step 4 to after step 7.

Purpose: This test insures that Recipient_List property value is maintained through a device "restart".

Test Concept: Initialize the Recipient_List property with a known value, then cycle power or restart the IUT and verify the Recipient_List property value is maintained.

Configuration Requirements: Base setup 1 for Notification Forwarder object tests.

Test Steps:

1. MAKE (Recipient_List = {(all), -- Valid Days
(all), -- From Time, To Time
DEST_OBJ_ID, -- Recipient D1
DEST_PROCESS_ID, -- Any Process Identifier
DEST_CONF_NOTIF, -- Any Issue Confirmed Notifications
{T, T, T} -- Transitions
}) -- One list element
2. IF (IUT supports the ReinitializeDevice service) THEN {
3. TRANSMIT ReinitializeDevice-Request,
'Reinitialized State of Device' = WARMSTART,
'Password' = (any valid password) --if required by IUT
4. RECEIVE BACnet-SimpleACK-PDU
~~}~~
~~*3*~~*5*. CHECK (Did the IUT perform a WARMSTART restart)
~~*4*~~*6*. WAIT for restart to complete
~~*5*~~*7*. VERIFY Recipient_List = {(all), -- Valid Days
(all), -- From Time, To Time
DEST_OBJ_ID, -- Recipient D1
DEST_PROCESS_ID, -- Process Identifier
DEST_CONF_NOTIF, -- Issue Confirmed Notifications
{T, T, T} -- Transitions
} -- One list element
}
~~*6*~~*8*. MAKE (The IUT reset by cycling power)
~~*7*~~*9*. WAIT for restart to complete
~~*8*~~*10*. VERIFY Recipient_List = {(all), -- Valid Days
(all), -- From Time, To Time
DEST_OBJ_ID, -- Recipient D1
DEST_PROCESS_ID, -- Process Identifier
DEST_CONF_NOTIF, -- Issue Confirmed Notifications
{T, T, T} -- Transitions
} -- One list element

7.3.2.30.13.2 Subscribed_Recipients Persistence Test

Reason for Change: Move closing curly bracket from after step 3 to after step 6..

© 2026 by BACnet International. All rights reserved.

Purpose: This test insures that Subscribed_Recipients property values are maintained through a device “restart”.

Test Concept: Initialize the Subscribed_Recipients property with a known value, then cycle power to restart the IUT and verify the Subscribed_Recipients property value is maintained.

Configuration Requirements: Base setup 2 for Notification Forwarder object tests with TR lifetime sufficient for this test.

Note To Tester: Start_Up_TimeA or Start_Up_TimeB is the time in minutes required to restart the IUT.

Test Steps:

1. IF (IUT supports the ReinitializeDevice service) THEN {
2. TRANSMIT ReinitializeDevice-Request,
'Reinitialized State of Device' = WARMSTART,
'Password' = (any valid password) --if required by IUT
3. RECEIVE BACnet-SimpleACK-PDU
~~}~~
~~2.~~ CHECK (Did the IUT perform a WARMSTART restart)
~~3~~*4*. WAIT for restart to complete, making a note of the time to restart as Start_Up_TimeA
~~4~~*5*. VERIFY Subscribed_Recipients =
{DEST_OBJ_ID, -- Recipient D1
DEST_PROCESS_ID, -- Process Identifier
DEST_CONF_NOTIF, -- Issue Confirmed Notifications
TR1 -- Time Remaining where (TR–Start_Up_TimeA-1) <= TR1 <= TR
} -- One list element
}
~~5~~*6*. MAKE (The IUT reset by cycling power)
~~6~~*7*.–WAIT for restart to complete, making a note of the time to restart as Start_Up_TimeB
~~7~~*8*. VERIFY Subscribed_Recipients =
{DEST_OBJ_ID, -- Recipient D1
DEST_PROCESS_ID, -- Process Identifier
DEST_CONF_NOTIF, -- Issue Confirmed Notifications
TR1 -- Time Remaining where (TR–Start_Up_TimeA-Start_Up_TimeB-2) <= TR1 <= TR
} -- One list element

~~Note To Tester: Start_Up_TimeA or Start_Up_TimeB is the time in minutes required to restart the IUT.~~

7.3.2.34 Channel Object Tests

7.3.2.34.4 Write_Status Test

Reason For Change: Updated to test that when the List_Of_Object_Property_References property is set to empty, the Write_Status property shall remain set to IDLE.

Purpose: To verify that the Write_Status of the Channel object is IDLE when it's List_Of_Object_Property_References is empty and is IN_PROGRESS while the Channel object's Present_Value is being propagated, FAILED when propagation failed, and SUCCESSFUL when propagation succeeds.

Test Concept: The Channel object's List_Of_Object_Property_References is read and verified to be empty. Then, the Channel object's Write_Status is verified to be IDLE. Next, any valid value is written to the Channel object's Present_Value and Write_Status is verified to be IDLE still. An object property reference, R1, that the IUT cannot reach is set into the List_Of_Object_Reference and then any valid value is written to the Present_Value and the Write_Status is verified to be IN_PROGRESS. After the IUT determines that the referenced device is offline, the Write_Status is verified to be FAILED. ~~Finally, any valid reference~~ *Next, a valid object property reference*, R2, that will cause successful value propagation is set to

© 2026 by BACnet International. All rights reserved.

the List_Of_Object_Property_References and the test is repeated to verify that Write_Status becomes SUCCESSFUL. *Finally, the List_Of_Object_Property_References is set to empty, and the Write_Status is then verified to remain IDLE.*

Test Steps:

1. READ LEN = List_Of_Object_Property_References, ARRAY INDEX = 0

-- IDLE test
2. READ L = List_Of_Object_Property_References
3. VERIFY L = (empty)
4. VERIFY Write_Status = IDLE
5. WRITE Present_Value = (any valid value)
6. VERIFY Write_Status = IDLE

-- IN_PROGRESS and FAIL test
7. WRITE List_Of_Object_Property_References = (R1) -- write the whole array
8. WRITE Present_Value = (any valid value)
9. VERIFY Write_Status = IN_PROGRESS
10. WAIT (**Channel Write Fail Time** * LEN)
11. VERIFY Write_Status = FAILED

-- SUCCESSFUL test
12. WRITE List_Of_Object_Property_References = (R2) -- write the whole array
13. WRITE Present_Value = (any valid value)
14. WAIT (**Channel Write Fail Time** * LEN)
15. VERIFY Write_Status = SUCCESSFUL

-- IDLE test
16. WRITE List_Of_Object_Property_References = (empty array)
17. VERIFY Write_Status = IDLE

7.3.2.39 Lighting Output Object Tests

[Remove test 7.3.2.39.14]

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

© 2026 by BACnet International. All rights reserved.

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
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

7.3.2.39.5 Lighting Command Operation RAMP_TO Test

Reason for change: Wrong expected value for Tracking_Value in Step 25. Fix In_Progress state in Steps 10 and 24.

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

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

© 2026 by BACnet International. All rights reserved.

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 ramp 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 RAMP_TO command (operation, target-value, priority, ramp-rate)
5. WRITE Lighting_Command = (RAMP_TO, V1, PTY1, any valid rate)
6. WAIT Internal Processing Fail Time
7. VERIFY Priority_ Array = V1, ARRAY INDEX = PTY1
8. VERIFY Present_Value =V1

-- Check In_Progress while ramping up
9. VERIFY In_Progress = RAMP_ACTIVE

-- Make sure that Tracking_Value increases with the ramp-rate
10. WHILE (In_Progress ~~<> IDLE~~== *RAMP_ACTIVE*) DO {
11. VERIFY Tracking_Value > 0 *and* < V1
12. CHECK (Tracking_Value is increasing with the ramp-rate)
    }

-- When ramping 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 ramp 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 RAMP_TO command (operation, target-value, priority, ramp-rate)
19. WRITE Lighting_Command = (RAMP_TO, V2, PTY1, any valid rate)
20. WAIT Internal Processing Fail Time
21. VERIFY Priority_ Array = V2, ARRAY INDEX = PTY1
22. VERIFY Present_Value = V2

-- Check In_Progress while ramping up
23. VERIFY In_Progress = RAMP_ACTIVE,

-- Make sure that Tracking_Value decreases with the ramp-rate
24. WHILE (In_Progress ~~<> RAMP_ACTIVE~~== *RAMP_ACTIVE*) DO {
25. VERIFY Tracking_Value > *V2 and* < ~~0~~ *100*
~~26. VERIFY Tracking_Value > V2~~
~~27~~*26*. CHECK (Tracking_Value is decreasing with the ramp-rate)
    }

-- Check In_Progress and Tracking_Value
~~28~~*27*. VERIFY In_Progress = IDLE

© 2026 by BACnet International. All rights reserved.

~~29~~*28*. VERIFY Tracking_Value = V2

7.3.2.39.11 Transition Test

Reason for Change: Remove checking physical output.

Purpose: To verify that the Lighting Output object transitions using the configured function and transitions at the configured speed when Transition is set to either FADE or RAMP.

Test Concept: Setup a Lighting Output object, O1, to use fading or ramping as the default transition method. Present_Value is changed to V1 which is larger than the initial Present_Value, V0, so that the output will fade or ramp up. Halfway through the process, verify that Tracking_Value is approximately equal to the value halfway between V0 and V1. ~~The physical output shall also be verified that it is fading or ramping from V0 to V1.~~ When the process completes, verify that Tracking_Value reached V1. Repeat the process fading or ramping down from V1 to V2.

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. The Transition property is set to FADE or RAMP, Present_Value is V0 and In_Progress is IDLE.

To test FADE functionality, T is FADE, A is FADE_ACTIVE, W1 and W2 are (Default_Fade_Time / 2), and Default_Fade_Time is sufficiently large so as to allow the intermediate progress checks.

To Test RAMP functionality, T is RAMP, A is RAMP_ACTIVE, W1 is (((V1 – V0) / Default_Ramp_Rate) / 2), W2 is (((V1 – V2) / Default_Ramp_Rate) / 2), and Default_Ramp_Rate is sufficiently small so as to allow the intermediate progress checks.

Test Steps:

1. VERIFY Transition = T
2. VERIFY In_Progress = IDLE
3. V0 = READ Present_Value
4. WRITE Present_Value = V1, ARRAY INDEX = PTY1
5. VERIFY Present_Value = V1
6. WAIT W1
7. VERIFY Tracking_Value ~= (V1 + V0) / 2
8. VERIFY In_Progress = A

~~9. CHECK (the physical output is fading from V0 to V1)~~

-- *(the physical output is fading from V0 to V1)*

~~10~~*9*. WAIT W1
~~11~~*10*. VERIFY In_Progress = IDLE
~~12~~*11*. VERIFY Tracking_Value = V1
~~13~~*12*. WRITE Present_Value = V2, ARRAY INDEX = PTY1
~~14~~*13*. VERIFY Present_Value = V2
~~15~~*14*. WAIT W2
~~16~~*15*. VERIFY Tracking_Value ~= (V2 + V1) / 2
~~17~~*16*. VERIFY In_Progress = A

~~18. CHECK (the physical output is fading V1 to V2)~~

-- *(the physical output is fading V1 to V2)*

~~19~~*17*. WAIT W2
~~20~~*18*. VERIFY In_Progress = IDLE
~~21~~*19*. VERIFY Tracking_Value = V2

7.3.2.40 Access Door Object Tests

[Remove test 7.3.2.40.1]

© 2026 by BACnet International. All rights reserved.

[Remove test 7.3.2.40.3]

[Remove test 7.3.2.40.4]

7.3.2.40.6 Door_Unlock_Delay_Time Test

Reason for Change: Simplified test by removing output checks and only running if Lock_Status property exists. Remove checking physical output.

Purpose: To verify that when the Door_Unlock_Delay_Time property has a non-zero value, the output is delayed in unlocking when a PULSE_UNLOCK or EXTENDED_PULSE_UNLOCK is written to the Present_Value and not when UNLOCK is written.

Test Concept: When unlocking the door by writing PULSE_UNLOCK to the Present_Value of the Access Door object, it is verified that the door is still locked for the specified Door_Pulse_Time, then the door is unlocked. The same test is done for EXTENDED_PULSE_UNLOCK, but this time, it is verified that the door is still locked for the specified Door_Extended_Pulse_Time then the door is unlocked.

Configuration Requirements: ~~The IUT shall be configured with a door control output that can be observed during the test.~~ The Relinquish_Default shall have the value LOCK. All writes to the Present_Value shall be performed at a priority higher than any internal algorithms writing to this property. Door_Unlock_Delay_Time shall be set to a non-zero value which is sufficient to *capture the value of the Lock_Status*~~observe the delay and check the status of the lock~~. Out_Of_Service shall be set to FALSE. Prior to the test the Present_Value shall have the value LOCK and the IUT is in a state that would cause the door to be locked.

Test Steps:

-- Test PULSE_UNLOCK
1. WRITE Present_Value = PULSE_UNLOCK
2. WAIT (**Internal Processing Fail Time**)
3. BEFORE Door_Unlock_Delay_Time
~~4. IF (Lock_Status is present) THEN~~
~~5~~*4*. VERIFY Lock_Status = LOCKED
~~6. CHECK (that the door control output is in a state that would cause the door to be locked)~~
-- door control output is in a state that would cause the door to be locked
5. WAIT (Door_Unlock_Delay_Time)
~~7. IF (Lock_Status is present) THEN~~
~~8~~*6*. VERIFY Lock_Status = UNLOCKED
~~9. CHECK (that the door control output is in a state that would cause the door to be unlocked)~~
-- door control output is in a state that would cause the door to be unlocked
~~10~~*7*.WAIT (Door_Pulse_Time)
~~11~~*8*. VERIFY Present_Value = LOCK
~~12. IF (Lock_Status is present) THEN~~
~~13~~*9*.VERIFY Lock_Status = LOCKED
~~14. CHECK (that the door control output is in a state that would cause the door to be locked)~~
-- door control output is in a state that would cause the door to be locked

-- Test EXTENDED_PULSE_UNLOCK
~~15~~*10*. WRITE Present_Value = EXTENDED_PULSE_UNLOCK
~~16~~*11*. WAIT (**Internal Processing Fail Time**)
~~17~~*12*. BEFORE Door_Unlock_Delay_Time
~~18. IF (Lock_Status is present) THEN~~
~~19~~*13*. VERIFY Lock_Status = LOCKED
~~20. CHECK (that the door control output is in a state that would cause the door to be locked)~~
-- door control output is in a state that would cause the door to be locked
14. WAIT (Door_Unlock_Delay_Time)
~~21. IF (Lock_Status is present) THEN~~

© 2026 by BACnet International. All rights reserved.

~~22~~*15*. VERIFY Lock_Status = UNLOCKED
~~23. CHECK (that the door control output is in a state that would cause the door to be unlocked)~~
-- door control output is in a state that would cause the door to be unlocked
~~24~~*16*. WAIT (Door_Extended_Pulse_Time)
~~25~~*17*. VERIFY Present_Value = LOCK
~~26. IF (Lock_Status is present) THEN~~
18. VERIFY Lock_Status = LOCKED
~~28. CHECK (that the door control output is in a state that would cause the door to be locked)~~
-- door control output is in a state that would cause the door to be locked

-- Test UNLOCK
~~29~~*19*. WRITE Present_Value = UNLOCK
~~30~~*20*. WAIT (**Internal Processing Fail Time**)
~~31. IF (Lock_Status is present) THEN~~
~~32~~*21*. VERIFY Lock_Status = UNLOCKED
~~33. CHECK (that the door control output is in a state that would cause the door to be unlocked)~~
-- door control output is in a state that would cause the door to be unlocked

7.3.2.40.X1 Present_Value and Lock_Status Test

Reason for Change: No test exists for this functionality.

Purpose: To verify that writing to the Present_Value will cause an appropriate change to Lock_Status.

Test Concept: The Present_Value property is written with each of the following values: UNLOCK, LOCK, PULSE_UNLOCK, EXTENDED_PULSE_UNLOCK and the Lock_Status property is monitored.

Configuration Requirements: The Relinquish_Default shall have the value LOCK. All writes are at a priority higher than any internal algorithms writing to this property. Out_Of_Service shall be set to FALSE. Prior to the test the Present_Value shall have the value LOCK and the IUT is in a state that would cause the door to be locked. Door_Pulse_Time and Door_Extended_Pulse_Time shall be configured with values large enough to allow verification of Lock_Status when the door is in an unlocked state.

Test Steps:

-- Test UNLOCK value
1. WRITE Present_Value = UNLOCK
2. WAIT (Internal Processing Fail Time)
3. VERIFY Lock_Status = UNLOCKED

-- Test LOCK value
4. WRITE Present_Value = LOCK
5. WAIT (Internal Processing Fail Time)
6. VERIFY Lock_Status = LOCKED

-- Test PULSE_UNLOCK value
7. WRITE Present_Value = PULSE_UNLOCK
8. WAIT (Internal Processing Fail Time)
9. WAIT (Door_Unlock_Delay_Time if present)
10. VERIFY Lock_Status = UNLOCKED
11. WAIT (Door_Pulse_Time)
12. VERIFY Present_Value = LOCK
13. VERIFY Lock_Status = LOCKED

-- Test EXTENDED_PULSE_UNLOCK value

© 2026 by BACnet International. All rights reserved.

14. WRITE Present_Value = EXTENDED_PULSE_UNLOCK
15. WAIT (Internal Processing Fail Time)
16. WAIT (Door_Unlock_Delay_Time if present)
17. VERIFY Lock_Status = UNLOCKED
18. WAIT (Door_Extended_Pulse_Time)
19. VERIFY Present_Value = LOCK
20. VERIFY Lock_Status = LOCKED

7.3.2.41 Access Point Object Tests

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_PROVISIONED = C20
u. DENIED_CREDENTIAL_LOCKOUT = C21
v. DENIED_CREDENTIAL_MAX_DAYS = C22
w. DENIED_CREDENTIAL_MAX_USES = C23
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 {

© 2026 by BACnet International. All rights reserved.

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

7.3.2.41.4 Authorization Mode Test

Reason for Change: Remove testing for NONE Authorization_Mode.

Purpose: To verify each authorization mode supported by this IUT.

Test Concept: For each authorization mode supported by the IUT, a valid credential is presented at the access point to verify that the appropriate action is taken.

Configuration Requirements: See Clause 7.3.2.41. This test requires the following additional configuration:

a) An active credential with valid access rights for the access point shall be represented by Access Credential object C1.
b) An active credential with no valid access rights shall be represented by Access Credential object C2.

Note: If the VERIFICATION_REQUIRED or AUTHORIZATION_DELAYED mode is supported, the vendor must provide a mechanism for external verification to be performed.

Test Steps:

-- verify GRANT_ACTIVE mode
1. IF (GRANT_ACTIVE is supported) THEN
READ EventTag = Access_Event_Tag
WRITE Authorization_Mode = GRANT_ACTIVE
MAKE (present credential C2 at credential reader for this access point)
VERIFY Access_Event = GRANTED
VERIFY Access_Event_Time = (the time that credential C2 was presented)
VERIFY Access_Event_Credential = C2
VERIFY Access_Event_Tag = EventTag + 1

-- verify DENY_ALL mode
2. IF (DENY_ALL is supported) THEN
READ EventTag = Access_Event_Tag
WRITE Authorization_Mode = DENY_ALL
MAKE (present credential C1 at credential reader for this access point)
VERIFY Access_Event = DENIED_DENY_ALL
VERIFY Access_Event_Time = (the time that credential C1 was presented)
VERIFY Access_Event_Credential = C1
VERIFY Access_Event_Tag = EventTag + 1

© 2026 by BACnet International. All rights reserved.

-- verify VERIFICATION_REQUIRED mode (verification authorized)
3. IF (VERIFICATION_REQUIRED is supported) THEN
READ EventTag = Access_Event_Tag
WRITE Authorization_Mode = VERIFICATION_REQUIRED
MAKE (present credential C1 at credential reader for this access point)
VERIFY Access_Event = VERIFICATION_REQUIRED
VERIFY Access_Event_Time = (the time that credential C1 was presented most recently)
VERIFY Access_Event_Credential = C1
VERIFY Authentication_Status = WAITING_FOR_VERIFICATION
MAKE (external verification process grants access)
VERIFY Access_Event = GRANTED
VERIFY Access_Event_Time = (the time that verification process granted access)
VERIFY Access_Event_Credential = C1
VERIFY Access_Event_Tag = EventTag + 1

-- verify VERIFICATION_REQUIRED mode (verification denied)
READ EventTag = Access_Event_Tag
WRITE Authorization_Mode = VERIFICATION_REQUIRED
MAKE (present credential C1 at credential reader for this access point)
VERIFY Access_Event = VERIFICATION_REQUIRED
VERIFY Access_Event_Time = (the time that credential C1 was presented)
VERIFY Access_Event_Credential = C1
VERIFY Authentication_Status = WAITING_FOR_VERIFICATION
MAKE (external verification process denies access)
VERIFY Access_Event = DENIED_VERIFICATION_FAILED
VERIFY Access_Event_Time = (the time that verification process denied access)
VERIFY Access_Event_Credential = C1
VERIFY Access_Event_Tag + 1

-- verify VERIFICATION_REQUIRED mode (verification timeout)
READ EventTag = Access_Event_Tag
WRITE Authorization_Mode = VERIFICATION_REQUIRED
MAKE (present credential C1 at credential reader for this access point)
VERIFY Access_Event = VERIFICATION_REQUIRED
VERIFY Access_Event_Time = (the time that credential C1 was presented)
VERIFY Access_Event_Credential = C1
VERIFY Authentication_Status = WAITING_FOR_VERIFICATION
MAKE (external verification process does not respond within verification time)
WAIT Verification_Time
VERIFY Access_Event = DENIED_VERIFICATION_TIMEOUT
VERIFY Access_Event_Time = (the time that verification process timed out)
VERIFY Access_Event_Credential = C1
VERIFY Access_Event_Tag = EventTag + 1

-- verify AUTHORIZATION_DELAYED mode (access granted)
4. IF (AUTHORIZATION_DELAYED is supported) THEN
WRITE Authorization_Mode = AUTHORIZATION_DELAYED
MAKE (present credential C1 at credential reader for this access point)
VERIFY Access_Event = AUTHORIZATION_DELAYED
VERIFY Access_Event_Time = (the time that credential C1 was presented)
VERIFY Access_Event_Credential = C1
MAKE (external verification process does not respond within verification time)
WAIT Verification_Time
VERIFY Access_Event = GRANTED
VERIFY Access_Event_Time = (the time that verification process timed out)
VERIFY Access_Event_Credential = C1

© 2026 by BACnet International. All rights reserved.

-- verify AUTHORIZATION_DELAYED mode (access denied)
WRITE Authorization_Mode = AUTHORIZATION_DELAYED
READ EventTag = Access_Event_Tag
MAKE (present credential C1 at credential reader for this access point)
VERIFY Access_Event = AUTHORIZATION_DELAYED
VERIFY Access_Event_Time = (the time that credential C1 was presented)
VERIFY Access_Event_Credential = C1
MAKE (external verification process denies access)
VERIFY Access_Event = DENIED_VERIFICATION_FAILED
VERIFY Access_Event_Time = (the time that verification process denied access)
VERIFY Access_Event_Credential = C1
VERIFY Access_Event_Tag = EventTag + 1

-- verify NONE mode - *no tests to perform*
~~5. IF (NONE is supported) THEN~~
~~WRITE Authorization_Mode = NONE~~
~~WAIT **Internal Processing Fail Time**~~
~~VERIFY Authentication_Status = DISABLED~~

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_Attempt_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)
9. VERIFY Access_Event_Credential = C1

© 2026 by BACnet International. All rights reserved.

-- 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)

7.3.2.41.10 Denied Access Disabled Credential Test

Reason for Change: Added object references for all properties in the test steps to improve readability.

Purpose: To test that a credential is denied access at an access point when the credential is disabled even when it has valid access rights.

Test Concept: A disabled credential is presented at an access point and access is denied. For testing purposes, the credential shall be disabled by setting the Expiration_Time to a time in the past.

Configuration Requirements: See Clause 7.3.2.41. This test requires the following additional configuration:
a) An access credential with valid access to AP1 shall be represented by Access Credential C1.
b) The Credential_Status property shall have the value ACTIVE.
c) The Reason_For_Disable property shall be empty.

Test Steps:

– test granted access when credential is active
1. VERIFY *C1,*Reason_For_Disable = ( )
2. VERIFY *C1,*Credential_Status = ACTIVE
3. MAKE (present credential C1 at credential reader for access point AP1)
4. VERIFY AP1,Access_Event = GRANTED

© 2026 by BACnet International. All rights reserved.

5. VERIFY AP1,Access_Event_Time = (the time that the credential C1 was presented)
6. VERIFY AP1,Access_Event_Credential = C1

– test denied access when credential is inactive
7. MAKE (*C1,*Expiration_Time = time < current time)
8. VERIFY *C1,*Reason_For_Disable = (DISABLED_EXPIRED)
9. VERIFY *C1,*Credential_Status = INACTIVE
10. MAKE (present credential C1 at credential reader for access point AP1)
11. VERIFY AP1,Access_Event = DENIED_CREDENTIAL_EXPIRED
12. VERIFY AP1,Access_Event_Time = (the time that the credential C1 was presented)
13. VERIFY AP1,Access_Event_Credential = C1

7.3.2.41.X1 Resizable Authentication Policy Properties Test

Reason for Change: New Test.

Purpose: This test verifies that when the size of an authentication policy property is changed, the size of the other authentication properties changes to match.

Test Concept: Change the size of each authentication policy property and verify the size of the other properties also changes to match.

Configuration Requirements: None.

Test Steps:

1. READ N1 = Number_Of_Authentication_Policies
2. IF (Number_Of_Authentication_Policies is writable) THEN
3. WRITE Number_Of_Authentication_Policies = (N2, any valid value other than N1)
   ELSE
4. MAKE (Number_Of_Authentication_Policies = (N2, any valid value other than N1)
5. IF (Authentication_Policy_List is present) THEN
6. VERIFY Authentication_Policy_List = N2, ARRAY INDEX = 0
7. IF (Authentication_Policy_Names is present) THEN
8. VERIFY Authentication_Policy_Names = N2, ARRAY INDEX = 0
9. IF (Authentication_Policy_List is present and its size can be changed) THEN {
10. READ N1 = Authentication_Policy_List, ARRAY INDEX = 0
11. IF (The size of Authentication_Policy_List is writable) THEN
12. WRITE Authentication_Policy_List = (N2, any valid value other than N1), ARRAY INDEX = 0
    ELSE
13. MAKE (The size of Authentication_Policy_List = N2, (any valid value other than N1))
14. VERIFY Number_Of_Authentication_Policies = N2
15. IF (Authentication_Policy_Names is present) THEN
16. VERIFY Authentication_Policy_Names = N2, ARRAY INDEX = 0
    }
17. IF (Authentication_Policy_Names is present and its size can be changed) THEN {
18. READ N1 = Authentication_Policy_Names, ARRAY INDEX = 0
19. IF (The size of Authentication_Policy_Names is writable) THEN
20. WRITE Authentication_Policy_Names = (N2, any valid value other than N1), ARRAY INDEX = 0
    ELSE
21. MAKE (The size of Authentication_Policy_Names = (N2, any valid value other than N1))
22. VERIFY Number_Of_Authentication_Policies = N2
23. IF (Authentication_Policy_List is present) THEN
24. VERIFY Authentication_Policy_List = N2, ARRAY INDEX = 0
    }

© 2026 by BACnet International. All rights reserved.

7.3.2.42 Access Zone Object Tests

7.3.2.42.4 Passback Mode Test*s*

Reason for Change: Replace single Passback Mode Test with smaller individual tests for clarity.

~~Purpose: To verify the passback functionality and passback exemption.~~

~~Test Concept: A valid credential is presented at the entry access point to this access zone. When the credential is presented a second time, a passback notification should be generated, and if the passback mode is set to hard passback, then access to the zone should be denied. If the credential has a passback exemption, then access will never be denied due to a passback violation.~~

~~Configuration Requirements: See Clause 7.3.2.42. This test requires the following additional configuration:~~

- ~~a) The Access Zone shall not be out of service.~~
- ~~b) An access point which is an entry point to this zone shall be represented by Access Point object AP1. The Authoriztion_Mode property of AP1 shall have the value Authorize.~~
- ~~c) An access point which is an exit point to this zone shall be represented by Access Point object AP2. The Authoriztion_Mode property of AP2 shall have the value Authorize.~~
- ~~d) An active credential with valid access rights for the access point AP1 and AP2 shall be represented by Access Credential object C1.~~
- ~~e) The C1.Authorization_Exemptions list shall be empty.~~

~~Test Steps:~~

~~- verify soft passback mode~~
~~1. MAKE (Passback_Mode = SOFT_PASSBACK)~~
~~2. READ EventTag = AP1,Access_Event_Tag~~
~~3. MAKE (present credential C1 at credential reader for access point AP1)~~
~~4. VERIFY AP1, Access_Event = GRANTED~~
~~5. VERIFY AP1,Access_Event_Time = (the time that credential C1 was presented)~~
~~6. VERIFY AP1,Access_Event_Credential = C1~~
~~7. VERIFY AP1,Access_Event_Tag = (EventTag + 1)~~
~~8. MAKE (present credential C1 at credential reader for access point AP1)~~
~~9. VERIFY AP1,Access_Event = GRANTED~~
~~10. VERIFY (AP1.Access_Event_Time = (the time that credential C1 was presented most recently)~~
~~11. VERIFY AP1.Access_Event_Credential = C1~~
~~12. VERIFY AP1,Access_Event_Tag = (EventTag +2)~~
~~13. MAKE (present credential C1 at credential reader for access point AP2)~~

~~- verify hard passback mode~~
~~14. MAKE (Passback_Mode = HARD_PASSBACK)~~
~~15. READ EventTag = AP1,Access_Event_Tag~~
~~16. MAKE (present credential C1 at credential reader for access point AP1)~~
~~17. VERIFY AP1.Access_Event = GRANTED~~
~~18. VERIFY AP1.Access_Event_Time = (the time that credential C1 was presented most recently)~~
~~19. VERIFY AP1.Access_Event_Credential = C1~~
~~20. VERIFY AP1,Access_Event_Tag = EventTag + 1~~
~~21. MAKE (present credential C1 at credential reader for access point AP1)~~
~~22. VERIFY AP1.Access_Event = DENIED_PASSBACK~~
~~23. VERIFY AP1.Access_Event_Time = (the time that credential C1 was presented most recently)~~
~~24. VERIFY AP1.Access_Event_Credential = C1~~
~~25. VERIFY AP1,Access_Event_Tag = (EventTag + 2)~~

© 2026 by BACnet International. All rights reserved.

~~-- verify passback timeout~~
~~26. WAIT (Passback_Timeout)~~
~~27. MAKE (present credential C1 at credential reader for access point AP1)~~
~~28. VERIFY AP1.Access_Event = GRANTED~~
~~29. VERIFY (AP1.Access_Event_Time = the time that credential C1 was presented)~~
~~30. VERIFY (AP1.Access_Event_Credential = C1)~~

~~-- verify hard passback off~~
~~31. MAKE (Passback_Mode = PASSBACK_OFF)~~
~~32. READ EventTag = AP1,Access_Event_Tag~~
~~33. MAKE (present credential C1 at credential reader for access point AP1)~~
~~34. VERIFY AP1.Access_Event = GRANTED~~
~~35. VERIFY AP1.Access_Event_Time = (the time that credential C1 was presented most recently)~~
~~36. VERIFY AP1.Access_Event_Credential = C1~~
~~37. VERIFY AP1,Access_Event_Tag = EventTag + 1~~
~~38. MAKE (present credential C1 at credential reader for access point AP1)~~
~~39. VERIFY AP1.Access_Event = GRANTED~~
~~40. VERIFY AP1.Access_Event_Time = (the time that credential C1 was presented most recently)~~
~~41. VERIFY (AP1.Access_Event_Credential = C1~~
~~42. VERIFY AP1,Access_Event_Tag = (EventTag + 2)~~

~~-- verify passback exemption~~
~~43. MAKE (Passback_Mode = HARD_PASSBACK)~~
~~44. MAKE (C1, Authorization_Exemption = (PASSBACK))~~
~~45. READ EventTag = AP1,Access_Event_Tag~~
~~46. MAKE (present credential C1 at credential reader for access point AP1)~~
~~47. VERIFY AP1.Access_Event = GRANTED~~
~~48. VERIFY AP1.Access_Event_Time = (the time that credential C1 was presented most recently)~~
~~49. VERIFY AP1.Access_Event_Credential = C1~~
~~50. VERIFY AP1,Access_Event_Tag = EventTag + 1~~
~~51. MAKE (present credential C1 at credential reader for access point AP1)~~
~~52. VERIFY AP1.Access_Event = GRANTED~~
~~53. VERIFY AP1.Access_Event_Time = (the time that credential C1 was presented most recently)~~
~~54. VERIFY AP1.Access_Event_Credential = C1~~
~~55. VERIFY AP1,Access_Event_Tag = (EventTag + 2)~~

7.3.2.42.4.X1 PASSBACK_OFF Mode Test

Reason for Change: Add new test for PASSBACK_OFF Mode.

Purpose: To verify the functionality of the Passback_Mode property when set to PASSBACK_OFF mode.

Test Concept: A valid credential is presented at an entry access point of an access zone with Passback_Mode set to PASSBACK_OFF. When the same valid credential is presented again at the same entry access point, without an intervening authorized exit, then a GRANTED Access_Event notification is generated.

Configuration Requirements: See Clause 7.3.2.42. This test requires the following additional configuration:
a) The Access Zone object, AZO1, shall have its Out_Of_Service property set to FALSE.
b) An Access Point object, APO1, representing an entry access point for the Access Zone, shall exist.
c) The Authorization_Mode property of APO1 shall be set to Authorize.
d) A valid credential, VC1, shall exist and shall have access rights for APO1.
e) An Access Credential object, AC1, representing VC1, shall exist.

Test Steps:

© 2026 by BACnet International. All rights reserved.

1. MAKE (AZO1, Passback_Mode = PASSBACK_OFF)
2. READ EventTag = APO1, Access_Event_Tag
3. MAKE (present credential VC1 at the credential reader for access point APO1)
4. VERIFY APO1, Access_Event = GRANTED
5. VERIFY APO1, Access_Event_Time = (the most recent time credential VC1 was presented.)
6. VERIFY APO1, Access_Event_Credential = AC1
7. VERIFY APO1, Access_Event_Tag = EventTag + 1
8. MAKE (present credential VC1 at the credential reader for access point APO1)
9. VERIFY APO1, Access_Event = GRANTED
10. VERIFY APO1, Access_Event_Time = (the most recent time credential VC1 was presented.)
11. VERIFY APO1, Access_Event_Credential = AC1
12. VERIFY APO1, Access_Event_Tag = (EventTag + 2)

7.3.2.42.4.X2 SOFT_PASSBACK Mode Test

Reason for Change: Add new test for SOFT_PASSBACK Mode.

Purpose: To verify the functionality of the Passback_Mode property when set to SOFT_PASSBACK.

Test Concept: A valid credential is presented at an entry access point of an access zone with Passback_Mode set to SOFT_PASSBACK. When the same valid credential is presented again at the same entry access point, without an intervening authorized exit, a PASSBACK_DETECTED Access_Event notification is generated.

Configuration Requirements: See Clause 7.3.2.42. This test requires the following additional configuration:
a) The Access Zone object, AZO1, shall have its Out_Of_Service property set to FALSE.
b) An Access Point object, APO1, representing an entry access point for the Access Zone, shall exist.
c) The Authorization_Mode property of APO1 shall be set to Authorize.
d) A valid credential, VC1, shall exist and shall have access rights for APO1.
e) An Access Credential object, AC1, representing VC1, shall exist.

Test Steps:
1. MAKE (AZO1, Passback_Mode = SOFT_PASSBACK)
2. READ EventTag = APO1, Access_Event_Tag
3. MAKE (present credential VC1 at the credential reader for access point APO1)
4. VERIFY APO1, Access_Event = GRANTED
5. VERIFY APO1, Access_Event_Time = (the time credential VC1 was presented)
6. VERIFY APO1, Access_Event_Credential = AC1
7. VERIFY APO1, Access_Event_Tag = (EventTag + 1)
8. MAKE (present credential VC1 at the credential reader for access point APO1)
9. VERIFY APO1, Access_Event = PASSBACK_DETECTED
10. VERIFY APO1, Access_Event_Time = (the most recent time credential VC1 was presented.)
11. VERIFY APO1, Access_Event_Credential = AC1
12. VERIFY APO1, Access_Event_Tag = (EventTag +2)
13. IF (Passback_Timeout property is present AND Passback_Timeout > 0) THEN
14. WAIT (Passback_Timeout)
15. MAKE (present credential VC1 at the credential reader for access point APO1)
16. VERIFY APO1, Access_Event = GRANTED
17. VERIFY APO1, Access_Event_Time = (the time credential VC1 was presented)
18. VERIFY APO1, Access_Event_Credential = AC1
19. VERIFY APO1, Access_Event_Tag = (EventTag + 3)

© 2026 by BACnet International. All rights reserved.

7.3.2.42.4.X3 HARD_PASSBACK Mode Test

Reason for Change: Add new test for HARD_PASSBACK Mode.

Purpose: To verify the functionality of the Passback_Mode property when set to HARD_PASSBACK mode.

Test Concept: A valid credential is presented at an entry access point of an access zone with Passback_Mode set to HARD_PASSBACK. When the same valid credential is presented again at the same entry access point, without an intervening authorized exit, a DENIED_PASSBACK Access_Event notification is generated.

Configuration Requirements: See Clause 7.3.2.42. This test requires the following additional configuration:
a) The Access Zone object, AZO1, shall have its Out_Of_Service property set to FALSE.
b) An Access Point object, APO1, representing an entry access point for the Access Zone, shall exist.
c) The Authorization_Mode property of APO1 shall be set to Authorize.
d) A valid credential, VC1, shall exist and shall have access rights for APO1.
e) An Access Credential object, AC1, representing VC1, shall exist.

Test Steps:
1. MAKE (AZO1, Passback_Mode = HARD_PASSBACK)
2. READ EventTag = APO1, Access_Event_Tag
3. MAKE (present credential VC1 at the credential reader for access point APO1)
4. VERIFY APO1, Access_Event = GRANTED
5. VERIFY APO1, Access_Event_Time = (the most recent time credential VC1 was presented.)
6. VERIFY APO1, Access_Event_Credential = AC1
7. VERIFY APO1, Access_Event_Tag = EventTag + 1
8. MAKE (present credential VC1 at the credential reader for access point APO1)
9. VERIFY APO1, Access_Event = DENIED_PASSBACK
10. VERIFY APO1, Access_Event_Time = (the most recent time credential VC1 was presented.)
11. VERIFY APO1, Access_Event_Credential = AC1
12. VERIFY APO1, Access_Event_Tag = (EventTag + 2)
13. IF (Passback_Timeout property is present AND Passback_Timeout > 0) THEN
14. WAIT (Passback_Timeout)
15. MAKE (present credential VC1 at the credential reader for access point APO1)
16. VERIFY APO1, Access_Event = GRANTED
17. VERIFY APO1, Access_Event_Time = (the time credential VC1 was presented)
18. VERIFY APO1, Access_Event_Credential = AC1
19. VERIFY APO1, Access_Event_Tag = (EventTag + 3)

7.3.2.42.4.X4 Passback Exemption Test

Reason for Change: Add new test to verify Passback Exemption.

Purpose: To verify the functionality of Passback Exemption.

Test Concept: A valid credential with Authorization_Exemption set to PASSBACK, is presented at an entry access point of an access zone with Passback_Mode set to HARD_PASSBACK. When the same valid credential is presented again at the same entry access point, without an intervening authorized exit, a GRANTED Access_Event notification is generated.

Test Conditionality: This test shall be executed if at least one of the IUT's Access Credential objects support the Authorization_Exemption property.

Configuration Requirements: See Clause 7.3.2.42. This test requires the following additional configuration:
a) The Access Zone object, AZO1, shall have its Out_Of_Service property set to FALSE.
b) An Access Point object, APO1, representing an entry access point for the Access Zone, shall exist.

© 2026 by BACnet International. All rights reserved.

c) The Authorization_Mode property of APO1 shall be set to Authorize.
d) A valid credential, VC1, shall exist and shall have access rights for APO1.
e) An Access Credential object, AC1, representing VC1, shall exist.
f) The Authorization_Exemption property of AC1 shall contain VC1

Test Steps:
1. MAKE (Passback_Mode = HARD_PASSBACK)
2. MAKE (AC1, Authorization_Exemption = PASSBACK)
3. READ EventTag = APO1, Access_Event_Tag
4. MAKE (present credential VC1 at the credential reader for access point APO1)
5. VERIFY APO1, Access_Event = GRANTED
6. VERIFY APO1, Access_Event_Time = (the most recent time credential VC1 was presented.)
7. VERIFY APO1, Access_Event_Credential = AC1
8. VERIFY APO1, Access_Event_Tag = EventTag + 1
9. MAKE (present credential VC1 at the credential reader for access point APO1)
10. VERIFY APO1, Access_Event = GRANTED
11. VERIFY APO1, Access_Event_Time = (the most recent time credential VC1 was presented.)
12. VERIFY APO1, Access_Event_Credential = AC1
13. VERIFY APO1, Access_Event_Tag = (EventTag + 2)

7.3.2.46 Network Port Object Tests

7.3.2.46.1 Network Port Configuration Tests

[Remove test 7.3.2.46.1.5]

7.3.2.46.2 Network-Number-Is Updates Network_Number_Quality Test

Reason for Change: Removed unicast Network-Number-Is.

Purpose: To verify that Network_Number_Quality is updated when the IUT learns its Network_Number from Network-Number-Is.

Test Concept: Write 0 to Network_Number to set Network_Number_Quality to UNKNOWN. Send a Network-Number-Is message to the IUT indicating that the Network_Number is learned and verify that Network_Number_Quality changes to LEARNED. Send a Network-Number-Is message to the IUT indicating that the Network_Number is configured and verify that Network_Number_Quality changes to LEARNED_CONFIGURED. Write 0 to Network_Number and verify that Network_Number_Quality changes to UNKNOWN.

Configuration Requirements: Select a Network Port object, O1, which is enabled and has a writable Network_Number. Connect the TD to the network associated with Network Port O1. This test shall be skipped if the TD cannot be directly connected to the IUT's network.

Test Steps:

-- set network number quality to UNKNOWN
1. WRITE Network_Number = 0
2. TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = ACTIVATE_CHANGES,
   'Password' = (any valid password)
3. RECEIVE BACnet-SimpleACK-PDU
4. WAIT Activate Changes Fail Time
5. VERIFY Network_Number_Quality = UNKNOWN

-- make IUT learn the network number

© 2026 by BACnet International. All rights reserved.

6. TRANSMIT Network-Number-Is
   DESTINATION = LOCAL_BROADCAST~~+IUT~~,
   'Network Number' = (N1: any valid value)
   'Flag' = 0 -- learned
7. VERIFY Network_Number_Quality = LEARNED
8. VERIFY Network_Number = N1

-- make IUT learn the network number from a configure device
9. TRANSMIT Network-Number-Is
   DESTINATION = LOCAL_BROADCAST~~+IUT~~,
   'Network Number' = (N2: any valid value)
   'Flag' = 1 -- configured
10. VERIFY Network_Number_Quality = LEARNED_CONFIGURED
11. VERIFY Network_Number = N2

-- configure the IUT's network number *by writing the Network_Number property*
12. WRITE Network_Number = (N3: any valid value other than 0)
13. TRANSMIT ReinitializeDevice-Request,
    'Reinitialized State of Device' = ACTIVATE_CHANGES,
    'Password' = (any valid password)
14. RECEIVE BACnet-SimpleACK-PDU
15. WAIT Activate Changes Fail Time
16. VERIFY Network_Number_Quality = CONFIGURED

-- *test Network-Number-Is is ignored when Network_Number_Quality is CONFIGURED*
17. TRANSMIT Network-Number-Is
    DESTINATION = LOCAL_BROADCAST~~+IUT~~,
    'Network Number' = (N4: any valid value)
    'Flag' = 1 -- configured
18. VERIFY Network_Number_Quality = CONFIGURED
19. VERIFY Network_Number = N3

-- revert network number quality to UNKNOWN
20. WRITE Network_Number = 0
21. TRANSMIT ReinitializeDevice-Request,
    'Reinitialized State of Device' = ACTIVATE_CHANGES,
    'Password' = (any valid password)
22. RECEIVE BACnet-SimpleACK-PDU
23. WAIT Activate Changes Fail Time
24. VERIFY Network_Number_Quality = UNKNOWN

7.3.2.46.3 Network Port Command Tests

7.3.2.46.3.2 DISCARD_CHANGES Command Tests

7.3.2.46.3.2.2 DISCARD_CHANGES Command Failure Test

Reason for change: Incorrect error code.

Purpose: To verify that Network Port object responds to DISCARD_CHANGES commands when the command is not supported.

© 2026 by BACnet International. All rights reserved.

Test Concept: Attempt to command a Network Port which does not support the DISCARD_CHANGES. Verify that the attempt fails with an Error Class of PROPERTY and an error code of ~~VALUE_OUT_OF_RANGE~~*OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED*.

Configuration Requirements: Select a Network Port which supports writable properties that set the Changes_Pending property to TRUE.

Test Steps:

1. TRANSMIT WriteProperty-Request,
   'Object Identifier' = (the Network Port object),
   'Property Identifier' = (any writable property that results in Changes_Pending = TRUE),
   'Property Value' = (any valid value)
2. RECEIVE BACnet-SimpleACK-PDU
3. TRANSMIT WriteProperty-Request,
   'Object Identifier' = (the Network Port object),
   'Property Identifier' = Command,
   'Property Value' = DISCARD_CHANGES,
4. RECEIVE BACnet-Error-PDU
   'Error Class' = PROPERTY,
   'Error Code' = ~~VALUE_OUT_OF_RANGE~~*OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED*
5. VERIFY Command = IDLE

7.3.2.46.3.3 RENEW_FD_REGISTRATION Command Tests

7.3.2.46.3.3.2 RENEW_FD_REGISTRATION Command Failure Test

Reason for change: Update test to match the error code specified in 135-2020ci-3. Remove references of disabling the command because this functionality will be tested in a new BTL test.

Purpose: To verify that *the* Network Port object respond*s* to *the* RENEW_FD_REGISTRATION command~~s~~ when the command is not supported. ~~/ enabled.~~

Test Concept: Attempt to command a Network Port which ~~is either not an BACnet/IP nor BACnet/IPv6 port or which is not in FOREIGN mode, to renew its FD subscription.~~*does not support RENEW_FD_REGISTRATION.* Verify that the attempt fails with *the correct error.* ~~an Error Class of PROPERTY and an error code of VALUE_OUT_OF_RANGE.~~

Configuration Requirements: Select a Network Port ~~which is not in FOREIGN mode~~ *which does not support RENEW_FD_REGISTRATION. Set the Network Port object in service.* If the IUT does not support the Command property, then this test shall be skipped.

Test Steps:

~~- make sure our initial conditions are good~~
~~1. IF Network_Type is IPV4 or IPV6 THEN~~
~~2. VERIFY BACnet_IP_Mode <> FOREIGN~~
1. VERIFY Out_Of_Service = FALSE
~~3~~*2*. TRANSMIT WriteProperty-Request,
   'Object Identifier' = (the Network Port object),
   'Property Identifier' = Command,
   'Property Value' = RENEW_FD_~~SUBSCRIPTION~~*REGISTRATION*,
~~4~~*3*. RECEIVE BACnet-Error-PDU
   'Error Class' = PROPERTY,
   'Error Code' = VALUE_OUT_OF_RANGE | *OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED*
~~5~~*4*. VERIFY Command = IDLE

© 2026 by BACnet International. All rights reserved.

7.3.2.46.3.3.X1 RENEW_FD_REGISTRATION Command Disable Test

Reason for change: New test.

Purpose: To verify that Network Port objects respond to the RENEW_FD_REGISTRATION command with the correct error codes when the command is disabled.

Test Concept: Start with a Network Port object that supports the RENEW_FD_REGISTRATION command. Attempt to write RENEW_FD_REGISTRATION to the Command property and verify that the device returns the correct error.

Configuration Requirements: The test is run on a Network Port object that supports the RENEW_FD_REGISTRATION command. The Network Port object is in service. For BACnet/IP, set BACnet_IP_Mode to be a value other than FOREIGN. For BACnet/IPv6, set BACnet_IPv6_Mode to be a value other than FOREIGN.

Test Steps:

1. VERIFY Out_Of_Service = False
2. TRANSMIT WriteProperty-Request,
   'Object Identifier' = (the Network Port object),
   'Property Identifier' = Command,
   'Property Value' = RENEW_FD_REGISTRATION,
3. RECEIVE BACnet-Error-PDU
   'Error Class' = PROPERTY,
   'Error Code' = NOT_ENABLED | VALUE_OUT_OF_RANGE
4. VERIFY Command = IDLE

7.3.2.46.3.4 RESTART_SUBORDINATE_DISCOVERY Command Tests

7.3.2.46.3.4.1 RESTART_~~SLAVE~~*SUBORDINATE*_DISCOVERY Command Test

Reason for Change: Fix MS/TP renaming oversights (RESTART_SLAVE_DISCOVERY is now RESTART_SUBORDINATE_DISCOVERY).

Purpose: To verify that the Network Port restarts the subordinate discovery process when commanded to.

Test Concept: Starting with a Network Port object which is configured as an MS/TP Subordinate Proxy, command the Network Port to *RESTART_SUBORDINATE_DISCOVERY* ~~RESTART_SLAVE_DISCOVERY~~. Verify that the IUT restarts subordinate discovery.

Configuration Requirements: Configure an MSTP Network Port object to act as an MS/TP Subordinate Proxy with Auto_Subordinate_Discovery set to TRUE. Configure the TD to act as an MS/TP subordinate. Delay the start of the test until after the IUT has completed its initial subordinate confirmation. If the IUT does not support automatic subordinate discovery, or the IUT does not support the *RESTART_SUBORDINATE_DISCOVERY* ~~RESTART_SLAVE_DISCOVERY~~ command, then this test shall be skipped.

Notes to Tester: The IUT may interrogate the subordinate addresses in any order. The IUT is allowed to generate any other traffic during the test, including, and is not limited to reading property values from the devices it finds.

Test Steps:

-- make sure our initial conditions are good
1. VERIFY Network_Type = MSTP
2. VERIFY Subordinate_Proxy_Enable = TRUE

© 2026 by BACnet International. All rights reserved.

-- request the renewal, and wait for it to timeout
3. WRITE Command = *RESTART_SUBORDINATE_DISCOVERY* ~~RESTART_SLAVE_DISCOVERY~~
4. BEFORE **Subordinate Proxy Confirm Interval**
5. REPEAT addr=(all MS/TP addresses excluding the IUT's MAC address) DO {
6. RECEIVE DESTINATION=addr, SRC=IUT
ReadProperty-Request,
'Object Identifier' = (DEVICE,4194303),
'Property Identifier' = Protocol_Services_Supported
}
~~5~~*7*. VERIFY Command = IDLE

7.3.2.46.3.4.2 RESTART_~~SLAVE~~*SUBORDINATE*_DISCOVERY Command Failure Test

Reason for change: Update test to match the error code specified in 135-2020ci-3. Remove references of disabling the command because this functionality will be tested in a new BTL test.

Purpose: To verify that *the* Network Port object respond*s* to *the* RESTART_~~SLAVE~~*SUBORDINATE*_DISCOVERY command~~s~~ when the command is not supported. ~~/ enabled.~~

Test Concept: Attempt to command a Network Port which ~~is not acting as a~~*does not support* subordinate proxy to RESTART_~~SLAVE~~*SUBORDINATE*_DISCOVERY. Verify that the attempt fails with ~~an Error Class of PROPERTY and an error code of VALUE_OUT_OF_RANGE.~~*the correct error.*

Configuration Requirements: Select a Network Port which ~~is not configured to be a~~*does not support* subordinate proxy. ~~If the IUT supports subordinate proxy functionality, this test shall be skipped as the standard does not specify how the IUT should respond when subordinate proxy is supported but not enabled.~~ *The Network Port object must be in service.* If the IUT does not support the Command property, then this test shall be skipped.

Test Steps:

1. TRANSMIT WriteProperty-Request,
'Object Identifier' = (the Network Port object),
'Property Identifier' = Command,
'Property Value' = RESTART_~~SLAVE~~*SUBORDINATE_DISCOVERY*
2. IF Network_Type is MSTP THEN
RECEIVE BACnet-Error-PDU
'Error Class' = PROPERTY,
'Error Code' = OPTIONAL_FUNCTIONALITY_*NOT*_SUPPORTED
ELSE
3. RECEIVE BACnet-Error-PDU
'Error Class' = PROPERTY,
'Error Code' = VALUE_OUT_OF_RANGE | *OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED*
~~3~~*4*. VERIFY Command = IDLE

7.3.2.46.3.4.X1 RESTART_SUBORDINATE_DISCOVERY Command Disable Test

Reason for change: New test.

Purpose: To verify that Network Port objects respond to the RESTART_SUBORDINATE_DISCOVERY command with the correct error codes when the command is disabled.

Test Concept: Start with a Network Port object that supports the RESTART_SUBORDINATE_DISCOVERY command. Attempt to write RESTART_SUBORDINATE_DISCOVERY to the Command property and verify that the device returns the correct error.

© 2026 by BACnet International. All rights reserved.

Configuration Requirements: The test is run on a Network Port object that supports the RESTART_SUBORDINATE_DISCOVERY command. The Network Port object is in service. Set Subordinate_Proxy_Enable = False.

Test Steps:

1. VERIFY Out_Of_Service = False
2. TRANSMIT WriteProperty-Request,
   'Object Identifier' = (the Network Port object),
   'Property Identifier' = Command,
   'Property Value' = RESTART_SUBORDINATE_DISCOVERY,
3. RECEIVE BACnet-Error-PDU
   'Error Class' = PROPERTY,
   'Error Code' = NOT_ENABLED | VALUE_OUT_OF_RANGE
4. VERIFY Command = IDLE

7.3.2.46.3.5 RENEW_DHCP Command Tests

7.3.2.46.3.5.2 RENEW_DHCP Command Failure Test

Reason for change: Update test to match the error code specified in 135-2020ci-3. Remove references of disabling the command because this functionality will be tested in a new BTL test.

Purpose: To verify that *the* Network Port object responds to *the* RENEW_DHCP command~~s~~ when the command is not supported. ~~/ enabled.~~

Test Concept: Attempt to command a Network Port which ~~is either not an BACnet/IP nor BACnet/IPv6 port or which is not in configured for auto-addressing.~~*does not support DHCP or auto addressing.* Verify that the attempt fails with ~~an Error Class of PROPERTY and an error code of VALUE_OUT_OF_RANGE.~~*the correct error.*

Configuration Requirements: Select a Network Port which ~~is not an IP or IPv6 port setup for auto-addressing.~~*does not support DHCP or auto addressing. The Network Port must be in service.* If the IUT does not support the Command property, then this test shall be skipped.

Test Steps:

~~- make sure our initial conditions are good~~
~~1. IF Network_Type is IPV4 and IP_DHCP_Enable is present THEN~~
~~VERIFY IP_DHCP_ENABLE = FALSE~~
~~2. IF Network_Type is IPV6 and IPV6_Auto_Addressing_Enabled is present THEN~~
~~VERIFY IPV6_Auto_Addressing_Enabled = FALSE~~

~~3~~*1.* TRANSMIT WriteProperty-Request,
   'Object Identifier' = (the Network Port object),
   'Property Identifier' = Command,
   'Property Value' = RENEW_DHCP

~~4~~*2.* IF Network_Type is IPV4 or IPV6 THEN
   RECEIVE BACnet-Error-PDU
      'Error Class' = PROPERTY,
      'Error Code' = OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED
   ELSE
3.   RECEIVE BACnet-Error-PDU
      'Error Class' = PROPERTY,
      'Error Code' = VALUE_OUT_OF_RANGE | *OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED*

© 2026 by BACnet International. All rights reserved.

~~5~~*4*. VERIFY Command = IDLE

7.3.2.46.3.5.X1 RENEW_DHCP Command Disable Test

Reason for change: New test.

Purpose: To verify that Network Port objects respond to the RENEW_DHCP command with the correct error codes when the command is disabled.

Test Concept: Start with a Network Port object that supports the RENEW_DHCP command. Attempt to write RENEW_DHCP to the Command property and verify that the device returns the correct error.

Configuration Requirements: The test is run on a Network Port object that supports the RENEW_DHCP command and supports IP_DHCP_Enable = False (for IPv4) or IPv6_Auto_Addressing_Enabled = False (for IPv6). The Network Port object is in service. For B/IPv4, set IP_DHCP_Enable = False. For B/IPv6, set IPv6_Auto_Addressing_Enabled = False.

Test Steps:

1. VERIFY Out_Of_Service = False
2. TRANSMIT WriteProperty-Request,
   'Object Identifier' = (the Network Port object),
   'Property Identifier' = Command,
   'Property Value' = RENEW_DHCP,
3. RECEIVE BACnet-Error-PDU
   'Error Class' = PROPERTY,
   'Error Code' = NOT_ENABLED | VALUE_OUT_OF_RANGE
4. VERIFY Command = IDLE

7.3.2.46.3.6 RESTART_AUTONEGOTIATION Command Tests

7.3.2.46.3.6.2 RESTART_AUTONEGOTIATION Command Failure Test

Reason for Change: Update test to match the error code specified in 135-2020ci-3. Remove references of disabling the command because this functionality will be tested in a new BTL test.

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~~,~~. *C*~~c~~ommand it to restart auto-negotiation. Verify that the correct error code is returned.

Configuration Requirements: *Execute test on a Network Port object which does not support RESTART_AUTONEGOTIATION.* ~~If the network port support auto-negotiation, disable it.~~ *The Network Port object must be in service.* 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.~~
~~VERIFY Link_Speed_Auto-negotiate = TRUE~~

~~request the renewal, and wait for it to timeout~~
1. VERIFY Out_Of_Service = False

© 2026 by BACnet International. All rights reserved.

2. TRANSMIT WriteProperty-Request,
   'Object Identifier' = (the Network Port object),
   'Property Identifier' = Command,
   'Property Value' = RESTART_AUTONEGOTIATION
3. *RECEIVE BACnet-Error-PDU*
   'Error Class' = PROPERTY,
   'Error Code' = VALUE_OUT_OF_RANGE | OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED
4. *VERIFY Command = IDLE*

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

7.3.2.46.3.6.X1 RESTART_AUTONEGOTIATION Command Disable Test

Reason for change: New test.

Purpose: To verify that Network Port objects respond to the RESTART_AUTONEGOTIATION command with the correct error codes when the command is disabled.

Test Concept: Start with a Network Port object that supports the RESTART_AUTONEGOTIATION command. Attempt to write RESTART_AUTONEGOTIATION to the Command property and verify that the device returns the correct error.

Configuration Requirements: The test is run on a Network Port object that supports the RESTART_AUTONEGOTIATION command and supports Link_Speed_Autonegotiate being set to False. The Network Port object is in service. Set Link_Speed_Autonegotiate = False.

Test Steps:

1. VERIFY Out_Of_Service = False
2. TRANSMIT WriteProperty-Request,
   'Object Identifier' = (the Network Port object),
   'Property Identifier' = Command,
   'Property Value' = RESTART_AUTONEGOTIATION,
3. RECEIVE BACnet-Error-PDU
   'Error Class' = PROPERTY,
   'Error Code' = NOT_ENABLED | VALUE_OUT_OF_RANGE
4. VERIFY Command = IDLE

7.3.2.46.3.7 DISCONNECT Command Tests

7.3.2.46.3.7.2 DISCONNECT Command Failure Test

Reason for change: Update test to match the error code specified in 135-2020ci-3. Remove references of disabling the command because this functionality will be tested in a new BTL test. Add verify step for IDLE.

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

© 2026 by BACnet International. All rights reserved.

Test Concept: With a Network Port object for a network which does not support disconnection, command it to disconnect. Verify that the correct error code is returned.

Configuration Requirements: If the IUT does not support the Command property, or all Network Port object*s* support disconnection, then this test shall be skipped.

Test Steps:

1. TRANSMIT WriteProperty-Request,
   'Object Identifier' = (the Network Port object),
   'Property Identifier' = Command,
   'Property Value' = DISCONNECT
2. RECEIVE BACnet-Error-PDU
   'Error Class' = PROPERTY,
   'Error Code' = VALUE_OUT_OF_RANGE | *OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED*
3. *VERIFY Command = IDLE*

7.3.2.46.3.10 Certificate Configuration Tests

7.3.2.46.3.10.2 GENERATE_CSR_FILE Command Failure Test

Reason for Change: Remove ‘enabled’ in the Purpose because there is no method of disabling this command. Update expected error code to support products at PR < 24 (before GENERATE_CSR_FILE was introduced). Add verify step for IDLE.

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

Test Concept: With a Network Port object for a network which does not support GENERATE_CSR_FILE. Verify that the correct error code is returned.

Configuration Requirements: property Changes Pending = FALSE.

Test Steps:

1. TRANSMIT WriteProperty-Request,
   'Object Identifier' = (the Network Port object),
   'Property Identifier' = Command,
   'Property Value' = GENERATE_CSR_FILE
2. *IF (Protocol_Revision >= 24) THEN*
~~2~~3. RECEIVE BACnet-Error-PDU
   'Error Class' = PROPERTY,
   'Error Code' = OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED
   ELSE
4. *RECEIVE BACnet-Error-PDU*
   'Error Class' = PROPERTY,
   'Error Code' = OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED |
   VALUE_OUT_OF_RANGE
5. *VERIFY Command = IDLE*

7.3.2.46.3.10.4 VALIDATE_CHANGES Command Failure Test

Reason for Change: Remove ‘enabled’ in the Purpose because there is no method of disabling this command. Update expected error code to support products at PR < 24 (before VALIDATE_CHANGES was introduced). Add verify step for IDLE.

© 2026 by BACnet International. All rights reserved.

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

Test Concept: With a Network Port object for a network which does not support VALIDATE_CHANGES. Verify that the correct error code is returned.

Test Steps:

1. TRANSMIT WriteProperty-Request,
   'Object Identifier' = (the Network Port object),
   'Property Identifier' = Command,
   'Property Value' = VALIDATE_CHANGES
2. *IF (Protocol_Revision >= 24) THEN*
~~2~~3. RECEIVE BACnet-Error-PDU
   'Error Class' = PROPERTY,
   'Error Code' = OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED
   ELSE
4. *RECEIVE BACnet-Error-PDU*
   'Error Class' = PROPERTY,
   'Error Code' = OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED | VALUE_OUT_OF_RANGE
5. *VERIFY Command = IDLE*

7.3.2.46.3.X1 RESTART_DEVICE_DISCOVERY Command Tests

7.3.2.46.3.X1.1 RESTART_DEVICE_DISCOVERY Command Test for Non-MSTP Networks

Reason for Change: No tests exist for this functionality

Purpose: To verify the IUT correctly clears the Device_Address_Proxy_Table and restarts Device Address Proxying when it receives a RESTART_DEVICE_DISCOVERY Command.

Test Concept:
With the IUT configured to proxy the network associated with the Network Port object being tested (NP), D1 is added to the Device_Address_Proxy_Table. D1 is then removed from the network, and the RESTART_DEVICE_DISCOVERY command is sent to NP. The Command property is read to verify that it went to IDLE and the Device_Address_Proxy_Table is read to verify that it no longer contains D1. D1 is then added to the network and the RESTART_DEVICE_DISCOVERY command is sent to NP. The Device_Address_Proxy_Table is read to verify that contains D1.

Configuration Requirements: Device_Address_Proxy_Enable = True. NP is the proxied network's Network Port object.

Test Steps:

1. TRANSMIT
   DA = LOCAL BROADCAST
   SA = D1
   I-Am Request,
   'Device Identifier' = (D1's Device Identifier),
   'Max APDU Length Accepted' = (any valid value),
   'Segmentation Supported' = (any valid value),
   'Vendor Identifier' = (TD's Vendor_Identifier
2. WAIT (10 seconds)
3. VERIFY Device_Address_Proxy_Table = (a list of values that contains D1)

-- Remove D1 from the network

© 2026 by BACnet International. All rights reserved.

4. MAKE (Remove D1 from the network)
5. WRITE NP, Command = RESTART_DEVICE_DISCOVERY
6. WAIT Internal Processing Fail Time
7. VERIFY NP, Command = IDLE
8. VERIFY Device_Address_Proxy_Table = (an empty list) | (a list containing only the IUT)
9. WAIT (10 seconds)
10. VERIFY Device_Address_Proxy_Table = (an empty list) | (a list containing only the IUT)

--Add D1 to the network

11. MAKE (Add D1 to the network)
12. WRITE NP, Command = RESTART_DEVICE_DISCOVERY
13. WAIT Internal Processing Fail Time
14. VERIFY NP, Command = IDLE
15. TRANSMIT
    DA = LOCAL BROADCAST
    SA = D1
    I-Am Request,
    'Device Identifier' = (D1's Device Identifier),
    'Max APDU Length Accepted' = (any valid value),
    'Segmentation Supported' = (any valid value),
    'Vendor Identifier' = (The TD's Vendor Identifier)
16. WAIT (10 seconds)
17. VERIFY Device_Address_Proxy_Table = (a list of values that contains D1)

7.3.2.46.3.X1.2 RESTART_DEVICE_DISCOVERY Command Test for MSTP Networks

Reason for Change: No tests exist for this functionality.

Purpose: To verify the IUT clears the Device_Address_Proxy_Table and restarts manager and subordinate node discovery when commanded to RESTART_DEVICE_DISCOVERY.

Test Concept: With the IUT configured to proxy the MSTP network associated with the Network Port object being tested (NP), a manager device, D1 and subordinate device, D2 are added to the Device_Address_Proxy_Table. D1 and D2 are removed from the network, and the RESTART_DEVICE_DISCOVERY command is sent to NP. The command property is read to verify that it went to idle and the Device_Address_Proxy_Table is read to verify that it no longer contains D1 nor D2. D1 and D2 are then added to the network, and the RESTART_DEVICE_DISCOVERY command is sent to the NP. The command property is read to verify that it went to idle, the Device_Address_Proxy_Table is read to verify that contains both D1 and D2, and the Subordinate_Address_Binding is read to verify it contains D2.

Test Configuration: The proxied MSTP network corresponding to the network port being tested (NP) is configured with a manager device, D1 and a subordinate device, D2. NP is configured with Subordinate_Proxy_Enable = TRUE and Device_Address_Proxy_Enable = TRUE. D2 shall be configured to support the special object instance of 4194303 and be configured with the Protocol_Services_Supported bit corresponding to the Who-Is service set to False. If the device supports auto subordinate discovery, Auto_Subordinate_Discovery shall be set to TRUE. Otherwise, D2 shall be present in the Manual_Subordinate_Address_Binding property at the start of the test. The vendor shall supply a maximum subordinate discovery time, DT.

Test Steps:

--make sure our initial conditions are good

1. VERIFY Network_Type = MSTP
2. VERIFY Subordinate_Proxy_Enable = TRUE
3. VERIFY Device_Address_Proxy_Enable = TRUE
4. TRANSMIT
    DA = LOCAL BROADCAST
    SA = D1
    I-Am Request,

© 2026 by BACnet International. All rights reserved.

'Device Identifier' = (D1's Device Identifier),
'Max APDU Length Accepted' = (any valid value),
'Segmentation Supported' = (any valid value),
'Vendor Identifier' = (TD's Vendor_Identifier)

5. WAIT Internal Processing Fail Time + 10 seconds
6. VERIFY Device_Address_Proxy_Table = (a list containing D1)
7. WAIT (for subordinate discovery to complete)
8. VERIFY Device_Address_Proxy_Table (a list containing both D1 and D2)
9. VERIFY NP, Subordinate_Address_Binding = (a list containing D2)
10. MAKE (Remove D1 and D2 from the network)
11. WRITE NP, Command = RESTART_DEVICE_DISCOVERY
12. WAIT Internal Processing Fail Time
13. VERIFY NP, Command = IDLE
14. VERIFY Device_Address_Proxy_Table = (a list that contains neither D1 nor D2)
15. MAKE (Add D1 and D2 to the network)
16. WAIT (time sufficient for D1 and D2 to join the network)
17. WRITE NP, Command = RESTART_DEVICE_DISCOVERY
18. WAIT Internal Processing Fail Time
19. VERIFY NP, Command = IDLE

--make sure devices are automatically added

20. TRANSMIT
    DA = LOCAL BROADCAST
    SA = D1
    I-Am Request,
    'Device Identifier' = (D1's Device Identifier),
    'Max APDU Length Accepted' = (any valid value),
    'Segmentation Supported' = (any valid value),
    'Vendor Identifier' = (TD's Vendor_Identifier)

21. WAIT (for manager and subordinate discovery to complete or DT)
22. VERIFY Device_Address_Proxy_Table (a list containing both D1 and D2)
23. VERIFY NP, Subordinate_Address_Binding = (a list containing D2)

7.3.2.46.3.X1.3 RESTART_DEVICE_DISCOVERY Command Failure Test

Reason for Change: No tests exist for this functionality

Purpose: To verify that the Network Port object responds to RESTART_DEVICE_DISCOVERY commands when the command is not supported.

Test Concept:

Attempt to command a Network Port which does not support Device Proxying or Subordinate Proxying to RESTART_DEVICE_DISCOVERY. Verify that the attempt fails with an Error Class of PROPERTY and an Error Code of OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED or VALUE_OUT_OF_RANGE.

Configuration Requirements: Select a Network Port which is not configured to support Device Proxying or Subordinate Proxying. Set the Network Port in service. If the IUT does not support the Command property, then this test shall be skipped.

Test Steps:

1. VERIFY Out_Of_Service = FALSE
2. TRANSMIT WriteProperty-Request
   'Object Identifier' = (The Network Port object),

© 2026 by BACnet International. All rights reserved.

'Property Identifier' = Command,
'Property Value' = RESTART_DEVICE_DISCOVERY
3. RECEIVE BACnet-Error-PDU
'Error Class' = PROPERTY,
'Error Code' = VALUE_OUT_OF_RANGE | OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED
4. VERIFY Command = IDLE

7.3.2.46.3.X1.X1 RESTART_DEVICE_DISCOVERY Command Disable Test

Reason for change: New test.

Purpose: To verify that Network Port objects respond to the RESTART_DEVICE_DISCOVERY command with the correct error codes when the command is disabled.

Test Concept: Start with a Network Port object that supports the RESTART_DEVICE_DISCOVERY command. Attempt to write RESTART_DEVICE_DISCOVERY to the Command property and verify that the device returns the correct error.

Configuration Requirements: The test is run on a Network Port object that supports the RESTART_DEVICE_DISCOVERY command and supports disabling the command. The Network Port object is in service. If Device_Address_Proxy_Enable is present, then set it to False. If Subordinate_Proxy_Enable is present, then set it to False.

Test Steps:

1. VERIFY Out_Of_Service = False
2. TRANSMIT WriteProperty-Request,
'Object Identifier' = (the Network Port object),
'Property Identifier' = Command,
'Property Value' = RESTART_DEVICE_DISCOVERY,
3. RECEIVE BACnet-Error-PDU
'Error Class' = PROPERTY,
'Error Code' = NOT_ENABLED | VALUE_OUT_OF_RANGE
4. VERIFY Command = IDLE

7.3.2.46.4 Hierarchical Network Port Tests

[Remove test 7.3.2.46.4.1]

7.3.2.46.7 DHCP Tests

7.3.2.46.7.1 Basic IPv4 DHCP Test

Reason for change: Fix step 1 indentation and renumber all steps.

Purpose: Verify that the IUT is able to participate in IPv4 DHCP and correctly report its DHCP status.

Test Concept: The DHCP server is removed from network. The IUT is then configured with an IPv4 network requiring DHCP, and if required, its DHCP settings are cleared. The related Network Port object is queried to verify that the DHCP related properties have the appropriate values indicating DHCP has not completed. The DHCP is connected to the network. It is verified that the IUT obtains network ~~setttings~~*settings* from the DHCP server, and that the DHCP properties reflect the current status.

Configuration Requirements: The DHCP is disconnected from the network or turned off. The IUT is configured for DHCP and any settings it previously received via DHCP are cleared.

Test Steps:

© 2026 by BACnet International. All rights reserved.

1. IF the IUT has a second enabled network port THEN {
2. VERIFY IP_DHCP_Enable = True
3. IF IP_DHCP_Lease_Time property is present THEN
4. VERIFY IP_DHCP_Lease_Time = 0
5. IF IP_DHCP_Lease_Time_Remaining property is present THEN
6. VERIFY IP_DHCP_Lease_Time_Remaining = 0
7. IF IP_DHCP_Server property is present THEN
8. VERIFY IP_DHCP_Server = X'00000000'
}
~~2~~*9*. MAKE(connect the DHCP server to the network)
~~3~~*10*.WAIT until the IUT obtains DHCP information
4*11*.IF IP_DHCP_Lease_Time property is present THEN
12. VERIFY IP_DHCP_Lease_Time = (0 or the value provided by the DHCP server)
~~5~~*13*.IF IP_DHCP_Lease_Time_Remaining property is present THEN
14. VERIFY IP_DHCP_Lease_Time_Remaining = (0 or a value less than that provided by the DHCP server)
~~6~~*15*.IF IP_DHCP_Server property is present THEN
16. VERIFY IP_DHCP_Server = (the DHCP server's address or X'00000000')
~~7~~*17*.VERIFY IP_Address = (the value served by the DHCP server)
~~8~~*18*.VERIFY IP_Default_Gateway = (the value served by the DHCP server)

7.3.2.46.7.2 Basic IPv6 DHCP Test

Reason for change: IPv6_DHCP_Enable property does not exist. Fix IPv6_DHCP_Server value.

Purpose: Verify that the IUT is able to participate in IPv6 DHCP and correctly report its DHCP status

Test Concept: The DHCP server is removed from network. The IUT is then configured with an IPv6 network requiring DHCP, and if required, its DHCP settings are cleared. The related Network Port object is queried to verify that the DHCP related properties have the appropriate values indicating DHCP has not completed. The DHCP is connected to the network. It is verified that the IUT obtains network settings from the DHCP server, and that the DHCP properties reflect the current DHCP status.

Configuration Requirements: The DHCP is disconnected from the network or turned off. The IUT is configured for DHCP and any settings it previously received via DHCP are cleared.

Test Steps:

1. IF the IUT has a second enabled network port THEN {
2. VERIFY ~~IPv6_DHCP_Enable~~*IPv6_Auto_Addressing_Enable* = True
3. IF IPv6_DHCP_Lease_Time property is present THEN
4. VERIFY IPv6_DHCP_Lease_Time = 0
5. IF IPv6_DHCP_Lease_Time_Remaining property is present THEN
6. VERIFY IPv6_DHCP_Lease_Time_Remaining = 0
7. IF IPv6_DHCP_Server property is present THEN
8. VERIFY IPv6_DHCP_Server = X'00000000*00000000000000000000000000000000*'
}
~~2~~*9*. MAKE (connect the DHCP server to the network)
~~3~~*10*.WAIT until the IUT obtains DHCP information
4*11*.IF IPv6_DHCP_Lease_Time property is present THEN
12. VERIFY IPv6_DHCP_Lease_Time = (0 or the value provided by the DHCP server)
~~5~~*13*.IF IPv6_DHCP_Lease_Time_Remaining property is present THEN
14. VERIFY IPv6_DHCP_Lease_Time_Remaining = (0 or a value less than that provided by the DHCP server)
~~6~~*15*.IF IPv6_DHCP_Server property is present THEN
16. VERIFY IPv6_DHCP_Server = (the DHCP server's address or X'00000000*00000000000000000000000000000000*')

© 2026 by BACnet International. All rights reserved.

~~7~~*17*. VERIFY IPv6_Address = (the value served by the DHCP server)
~~8~~*18*. VERIFY IPv6_Default_Gateway = (the value served by the DHCP server)

7.3.2.46.X Valid Hierarchy Tests

7.3.2.46.X.1 Valid Hierarchy Chain - MSTP Test

Reason for Change: Explicit test for this data link.

Purpose: To verify that a hierarchical chain of MSTP network port objects in the IUT is organized in a valid hierarchy of three protocol levels.

Test Concept: Starting with a hierarchical Network Port object (NP_APP) which represents a configured application layer port (BACNET_APPLICATION), verify this port contains valid Network_Type, Protocol_Level, and Reference_Port. Visit each Network Port object in the hierarchy ensuring that the Network_Type, Protocol_Level, and Reference_Port are valid.

Notes to Tester: NPOs of Network_Type = MSTP are excluded from this test if they are at Protocol_Level = PROTOCOL and they are not referenced by another NPO. NPOs are also excluded from this test if they are at Protocol_Level = PHYSICAL and Network_Type = SERIAL and they are not referenced by another NPO.

Test Steps:

-- Verify Protocol_Level = BACNET_APPLICATION
1. VERIFY NP_APP, Protocol_Level = BACNET_APPLICATION
2. VERIFY NP_APP, Network_Type = MSTP
3. VERIFY NP_APP, Reference_Port <> 4194303

-- Verify Protocol_Level = PROTOCOL
4. RP = READ (NP_APP, Reference_Port)
5. NP_PROT = (Network Port, RP)
6. VERIFY NP_PROT, Protocol_Level = PROTOCOL
7. VERIFY NP_PROT, Network_Type = MSTP
8. VERIFY NP_PROT, Reference_Port <> 4194303

-- Verify Protocol_Level = PHYSICAL
9. RP = READ (NP_PROT, Reference_Port)
10. NP_PHY = (Network Port, RP)
11. VERIFY NP_PHY, Protocol_Level = PHYSICAL
12. VERIFY NP_PHY, Network_Type = SERIAL
13. VERIFY NP_PHY, Reference_Port = 4194303

7.3.2.46.X.2 Valid Hierarchy Chain - IPvX Test

Reason for Change: Explicit test for this data link.

Purpose: To verify that a hierarchical chain of IPV4 or IPV6 network port objects in the IUT is organized in a valid hierarchy of protocol levels.

Test Concept: Starting with a hierarchical Network Port object (NP_APP) and Network_Type (NT) of IPV4 or IPV6 which represents a configured application layer port (BACNET_APPLICATION), verify this port contains valid Network_Type, Protocol_Level, and Reference_Port. Visit each Network Port object in the hierarchy ensuring that the Network_Type, Protocol_Level, and Reference_Port are valid.

© 2026 by BACnet International. All rights reserved.

Notes to Tester: NPOs of this Network_Type are excluded from this test if they are at Protocol_Level = PROTOCOL and they are not referenced by another NPO. NPOs are also excluded from this test if they are at Protocol_Level = PHYSICAL and Network_Type = ETHERNET and they are not referenced by another NPO.

Test Steps:

```
-- Verify Protocol_Level = BACNET_APPLICATION
1.   VERIFY NP_APP, Protocol_Level = BACNET_APPLICATION
2.   VERIFY NP_APP, Network_Type = NT
3.   RP = READ NP_APP, Reference_Port
4.   VERIFY RP <> 4194303

-- Verify required NPO at Protocol_Level = PROTOCOL
5.   NP_PROT = (Network Port, RP)
6.   VERIFY NP_PROT, Protocol_Level = PROTOCOL
7.   VERIFY NP_PROT, Network_Type = NT
8.   RP = READ NP_PROT, Reference_Port
9.   IF (RP <> 4194303) THEN
     {
          -- Check if Protocol_Level = PHYSICAL
10.       NPx = (Network Port, RP)
11.       IF (NPx, Protocol_Level = PHYSICAL) THEN
               -- PHYSICAL level. Could be ETHERNET or proprietary
12.            VERIFY NPx, Reference_Port = 4194303
          ELSE
          {
               -- Some number of NPOs at PROTOCOL level
               -- move down the chain of NPOs to the bottom
13.            WHILE (RP <> 4194303)
               {
14.                 NPx = (Network Port, RP)
15.                 VERIFY NPx, Protocol_Level <> (BACNET_APPLICATION OR NON_BACNET_APPLICATION)
                    -- check that if PHYSICAL level it is the last NPO
16.                 IF (NPx, Protocol_Level == PHYSICAL) THEN
17.                      VERIFY NPx, Reference_Port = 4194303
18.                 RP = READ NPx, Reference_Port
               }
               -- either the NPO is at PROTOCOL and does not reference another NPO or it is PHYSICAL
               -- If PHYSICAL, it could be ETHERNET or proprietary or SERIAL -- done
          }
     }
     ELSE
     {
          -- NPO is at PROTOCOL and does not reference another NPO - done
     }
```

7.3.2.46.X.3 Valid Hierarchy Chain - Application and Physical Test

Reason for Change: Explicit test for data links without PROTOCOL level NPOs.

Purpose: To verify that a hierarchical chain of network port objects in the IUT is organized in a valid hierarchy of BACNET_APPLICATION and PHYSICAL protocol levels.

© 2026 by BACnet International. All rights reserved.

Test Concept: Starting with a hierarchical Network Port object (NP_APP) which represents a configured application layer port-(BACNET_APPLICATION), verify this port contains the correct Network_Type (NT), Protocol_Level, and Reference_Port. Visit the next level ensuring that the Network_Type, Protocol_Level, and Reference_Port are correct.

Notes to Tester: NPOs of this Network_Type with Protocol_Level = PHYSICAL are excluded from this test if they are not referenced by another NPO.

Test Steps:

-- Verify Protocol_Level = BACNET_APPLICATION
1. VERIFY NP_APP, Protocol_Level = BACNET_APPLICATION
2. VERIFY NP_APP, Network_Type = NT
3. VERIFY NP_APP, Reference_Port <> 4194303

-- Verify Protocol_Level = PHYSICAL
4. RP = READ (NP_APP, Reference_Port)
5. NP_PHY = (Network Port, RP)
6. VERIFY NP_PHY, Protocol_Level = PHYSICAL
7. VERIFY NP_PHY, Network_Type = NT
8. VERIFY NP_PHY, Reference_Port = 4194303

7.3.2.46.X.4 Valid Hierarchy Chain - Secure Connect Test

Reason for Change: Explicit test for this data link.

Purpose: To verify that a hierarchical chain of SECURE_CONNECT network port objects in the IUT is organized in a valid hierarchy of protocol levels.

Test Concept: Starting with a hierarchical Network Port object (NP_APP) and Network_Type of SECURE_CONNECT which represents a configured application layer port (BACNET_APPLICATION), verify this port contains valid Network_Type, Protocol_Level, and Reference_Port. Visit each Network Port object in the hierarchy ensuring that the Network_Type, Protocol_Level and Reference_Port are valid. Using the Additional_Reference_Ports property, visit each Network Port object in the hierarchy ensuring that the Network_Type, Protocol_Level, Reference_Port, and Additional_Reference_Ports are valid.

Test Steps:

-- Verify Protocol_Level = BACNET_APPLICATION
1. VERIFY NP_APP, Protocol_Level = BACNET_APPLICATION
2. VERIFY NP_APP, Network_Type = SECURE_CONNECT
3. RP = READ NP_APP, Reference_Port
4. VERIFY RP <> 4194303

-- Check the rest of the chain
5. WHILE (RP <> 4194303)
   {
6. NPx = (Network Port, RP)
7. VERIFY NPx, Protocol_Level <> (NON_BACNET_APPLICATION or BACNET_APPLICATION)
8. VERIFY NPx, Network_Type <> SECURE_CONNECT
   -- check that if PHYSICAL level it is the last NPO
9. IF (NPx, Protocol_Level == PHYSICAL) THEN
10. VERIFY NPx, Reference_Port = 4194303
11. RP = READ NPx, Reference_Port
   }
   -- either the NPO is at PROTOCOL and does not reference another NPO or it is PHYSICAL

© 2026 by BACnet International. All rights reserved.

```
12.  IF (NP_APP, Additional_Reference_Ports is present) THEN {
13.       REPEAT ARP = (for each entry in NP_APP, Additional_Reference_Ports) DO {
14.            NPx = (Network Port, ARP)
15.            VERIFY NPx, Protocol_Level <> (NON_BACNET_APPLICATION or BACNET_APPLICATION)
16.            VERIFY NPx, Network_Type <> SECURE_CONNECT
               -- check that if PHYSICAL level it is the last NPO
17.            IF (NPx, Protocol_Level == PHYSICAL) THEN
18.                 IF (NPx, Additional_Reference_Ports is present) THEN
19.                      VERIFY NPx, Additional_Reference_Ports = (empty list)
               -- Check the rest of the chain
20.            RP = READ NPx, Reference_Port
21.            WHILE (RP <> 4194303)
               {
22.                 NPx = (Network Port, RP)
23.                 VERIFY NPx, Protocol_Level <> (NON_BACNET_APPLICATION
                         or BACNET_APPLICATION)
24.                 VERIFY NPx, Network_Type <> SECURE_CONNECT
                    -- check that if PHYSICAL level it is the last NPO
25.                 IF (NPx, Protocol_Level == PHYSICAL) THEN
26.                      VERIFY NPx, Reference_Port = 4194303
27.                 RP = READ NPx, Reference_Port
               }
          }
     }
```

7.3.2.46.X.5 Valid Hierarchy Chain - Proprietary Test

Reason for Change: Explicit test for this data link.

Purpose: To verify that a hierarchical chain of non-BACnet network port objects in the IUT is organized in a valid hierarchy of protocol levels.

Test Concept: Starting with a hierarchical Network Port object (NP_APP) and Network_Type (NT) which represents a configured application layer port (NON_BACNET_APPLICATION), verify this port contains valid Network_Type, Protocol_Level, and Reference_Port. Visit each Network Port object in the hierarchy ensuring that the Network_Type, Protocol_Level, and Reference_Port are valid. Using the Additional_Reference_Ports property, visit each Network Port object in the hierarchy ensuring that the Network_Type, Protocol_Level, Reference_Port, and Additional_Reference_Ports are valid.

Notes to Tester: NPOs of this Network_Type are excluded from this test if they are at Protocol_Level = PROTOCOL and they are not referenced by another NPO. NPOs of this Network_Type are also excluded from this test if they are at Protocol_Level = PHYSICAL and they are not referenced by another NPO.

Test Steps:

```
-- Verify Protocol_Level = NON_BACNET_APPLICATION
1.   VERIFY NP_APP, Protocol_Level = NON_BACNET_APPLICATION
2.   VERIFY NP_APP, Network_Type = NT
3.   RP = READ NP_APP, Reference_Port
4.   VERIFY RP <> 4194303

-- Check the rest of the chain
5.   WHILE (RP <> 4194303)
     {
```

© 2026 by BACnet International. All rights reserved.

```
6.        NPx = (Network Port, RP)
7.        VERIFY NPx, Protocol_Level <> (NON_BACNET_APPLICATION OR BACNET_APPLICATION)
          -- check that if PHYSICAL level it is the last NPO
8.        IF (NPx, Protocol_Level == PHYSICAL) THEN
9.             VERIFY NPx, Reference_Port = 4194303
10.       RP = READ NPx, Reference_Port
     }
     -- either the NPO is at PROTOCOL and does not reference another NPO or it is PHYSICAL

11.  IF (Protocol_Revision >= 24 and NP_APP, Additional_Reference_Ports is present) THEN {
12.       REPEAT ARP = (for each entry in NP_APP, Additional_Reference_Ports) DO {
13.            NPx = (Network Port, ARP)
14.            VERIFY NPx, Protocol_Level <> (NON_BACNET_APPLICATION or BACNET_APPLICATION)
               -- check that if PHYSICAL level it is the last NPO
15.            IF (NPx, Protocol_Level == PHYSICAL) THEN
16.                 IF (NPx, Additional_Reference_Ports is present) THEN
17.                      VERIFY NPx, Additional_Reference_Ports = (empty list)
               -- Check the rest of the chain
18.            RP = READ NPx, Reference_Port
19.            WHILE (RP <> 4194303)
               {
20.                 NPx = (Network Port, RP)
21.                 VERIFY NPx, Protocol_Level <> (NON_BACNET_APPLICATION
                         or BACNET_APPLICATION)
                    -- check that if PHYSICAL level it is the last NPO
22.                 IF (NPx, Protocol_Level == PHYSICAL) THEN
23.                      VERIFY NPx, Reference_Port = 4194303
24.                 RP = READ NPx, Reference_Port
               }
          }
     }
```

7.3.2.46.X.6 Valid Hierarchy Chain - Virtual Test

Reason for Change: Explicit test for this data link.

Purpose: To verify that a hierarchical chain of BACnet network port objects in the IUT is organized in a valid hierarchy of protocol levels.

Test Concept: Starting with a hierarchical Network Port object (NP_APP) and Network_Type which represents a configured application layer port (BACNET_APPLICATION), verify this port contains valid Network_Type, Protocol_Level, and Reference_Port. Visit each Network Port object in the hierarchy ensuring that the Network_Type, Protocol_Level, and Reference_Port are valid. Using the Additional_Reference_Ports property, visit each Network Port object in the hierarchy ensuring that the Network_Type, Protocol_Level, Reference_Port, and Additional_Reference_Ports are valid.

Notes to Tester: NPOs of this Network_Type are excluded from this test if they are at Protocol_Level = PROTOCOL and they are not referenced by another NPO. NPOs of this Network_Type are also excluded from this test if they are at Protocol_Level = PHYSICAL and they are not referenced by another NPO.

Test Steps:

```
-- Verify Protocol_Level = BACNET_APPLICATION
1.  VERIFY NP_APP, Protocol_Level = BACNET_APPLICATION
2.  VERIFY NP_APP, Network_Type = VIRTUAL
3.  RP = READ NP_APP, Reference_Port
```

© 2026 by BACnet International. All rights reserved.

```
4.  VERIFY RP <> 4194303

-- Check the rest of the chain
5.  WHILE (RP <> 4194303)
    {
6.      NPx = (Network Port, RP)
7.      VERIFY NPx, Protocol_Level <> (NON_BACNET_APPLICATION OR BACNET_APPLICATION)
        -- check that if PHYSICAL level it is the last NPO
8.      IF (NPx, Protocol_Level == PHYSICAL) THEN
9.          VERIFY NPx, Reference_Port = 4194303
10.     RP = READ NPx, Reference_Port
    }
    -- either the NPO is at PROTOCOL and does not reference another NPO or it is PHYSICAL

11. IF (Protocol_Revision >= 24 and NP_APP, Additional_Reference_Ports is present) THEN {
12.     REPEAT ARP = (for each entry in NP_APP, Additional_Reference_Ports) DO {
13.         NPx = (Network Port, ARP)
14.         VERIFY NPx, Protocol_Level <> (NON_BACNET_APPLICATION or BACNET_APPLICATION)
            -- check that if PHYSICAL level it is the last NPO
15.         IF (NPx, Protocol_Level == PHYSICAL) THEN
16.             IF (NPx, Additional_Reference_Ports is present) THEN
17.                 VERIFY NPx, Additional_Reference_Ports = (empty list)
            -- Check the rest of the chain
18.         RP = READ NPx, Reference_Port
19.         WHILE (RP <> 4194303)
            {
20.             NPx = (Network Port, RP)
21.             VERIFY NPx, Protocol_Level <> (NON_BACNET_APPLICATION
                        or BACNET_APPLICATION)
                -- check that if PHYSICAL level it is the last NPO
22.             IF (NPx, Protocol_Level == PHYSICAL) THEN
23.                 VERIFY NPx, Reference_Port = 4194303
24.             RP = READ NPx, Reference_Port
            }
        }
    }
```

7.3.2.47 Timer Object

7.3.2.47.1 Positive Tests

7.3.2.47.1.2 Timer Running then Expired Test

Reason for change: Changing test step order to improve verifying Update_Time against “the current date time” and remove unnecessary CHECKs.

Purpose: Verify that Timer T1 when set to RUNNING, enters the EXPIRED state after the configured time.

Test Concept: Start this test in the IDLE state. Write Timer_Running and observe it enter RUNNING state, with its current configuration. After time passes, observe it enters the EXPIRED state. Observe that specified properties take their required values.

Configuration Requirements: T1 starts this test with the Timer_State equal to IDLE.

© 2026 by BACnet International. All rights reserved.

Test Steps:

1. VERIFY Timer_State = IDLE
2. WRITE Timer_Running = TRUE
~~3. CHECK (IUT exhibits any changes configured in IDLE_TO_RUNNING transition)~~
~~4~~*3*. IF (Default_Timeout property is present in T1) THEN {
4. READ DV = Default_Timeout
5. VERIFY Initial_Timeout = DV
}
~~5~~*6*. VERIFY Timer_State = RUNNING
~~6~~*7*. READ RT = Present_Value
~~7~~*8*. WAIT RT
~~8. CHECK (IUT exhibits any changes configured in RUNNING_TO_EXPIRED transition)~~
9. VERIFY Timer_State = EXPIRED
~~10. VERIFY Present_Value = 0~~
~~11. VERIFY Last_State_Change = RUNNING_TO_EXPIRED~~
~~12~~*10*. IF (Update_Time property is present in T1) THEN {
11. READ UT = Update_Time
12. VERIFY UT ~= (the current date and time)
13. IF (Expiration_Time property is present in T1) THEN
14. VERIFY Expiration_Time = UT
}
ELSE *{*
15. IF (Expiration_Time property is present in T1) THEN
16. VERIFY Expiration_Time ~= (the current date and time)
}
17. VERIFY Present_Value = 0
18. VERIFY Last_State_Change = RUNNING_TO_EXPIRED

7.3.2.47.1.12 Forcing Timer Expiration by Writing IDLE

Reason for Change: In Step 7 verifying the exact Update_Time against the current date and time is not possible.

Purpose: Interrupting the Timer while it is RUNNING, via a value of IDLE written to the Timer_State property.

Test Concept: Configure and start the Timer T1 to operate according to its values. Then write IDLE to Timer_State and observe that specified properties take their required values and all configured State_Change_Values transitions if any, take place.

Configuration Requirements: T1 starts this test with the Timer_State equal to RUNNING.

Test Steps:
1. VERIFY Timer_Running = TRUE
2. VERIFY Timer_State = RUNNING
3. WRITE Timer_State = IDLE
4. CHECK (IUT exhibits any changes configured in RUNNING_TO_IDLE transition)
5. VERIFY Timer_State = IDLE
6. VERIFY Last_State_Change = RUNNING_TO_IDLE
7. IF (Expiration_Time property is present in T1) THEN
8. VERIFY Expiration_Time = (the unspecified datetime value)
~~8~~*9*. IF (Update_Time property is present in T1) THEN
~~VERIFY Update_Time = (the current date and time)~~
10. VERIFY Update_Time ~= (the current date and time)
~~9~~*11*. VERIFY Present_Value = 0

© 2026 by BACnet International. All rights reserved.

7.3.2.50 Staging Object Tests

7.3.2.50.11 Out_Of_Service, Status_Flags, and Reliability for Staging Object

Reason for Change: CONFIGURATION_ERROR added in step 13, it was previously omitted.

Purpose: To verify that Present_Value and Reliability are writable when Out_Of_Service is TRUE, to verify the relationship between Out_Of_Service, Status_Flags, and Reliability, and to verify that writes to Target_References only occur when Out_Of_Service is FALSE.

Test Concept: The Out_Of_Service property is set to TRUE and the value of the Status_Flags property is validated. Present_Value is modified to verify that Present_Stage evaluates but writes to Target_References do not occur. If the IUT supports Reliability values other than NO_FAULT_DETECTED, writability for that property is tested and the value of the Status_Flags property is validated. The Out_Of_Service property is set to FALSE and the value of the Status_Flags property is validated. The Present_Value for one of the Target_References is checked to verify that it has the correct value, indicative of a write that occurred when transitioning Out_Of_Service from TRUE to FALSE.

Configuration Requirements: The Staging object used for this test shall be configured with at least one object in the Target_References property. The Stages property shall be configured with two stages such that Stages[S].Values = {V1…} and Stages[S+1].Values = {V2…} where V1 <> V2. At the start of the test, the Staging object is properly configured such that Reliability = NO_FAULT_DETECTED and Present_Stage = S.

Test Steps:

1. READ SF1 = Status_Flags
2. VERIFY Reliability = NO_FAULT_DETECTED
3. VERIFY Present_Stage = S
4. READ O1 = Target_References, ARRAY INDEX = 1
5. VERIFY O1, Present_Value = V1
6. IF (Out_Of_Service is writable) THEN
   WRITE Out_Of_Service = TRUE
   ELSE
   MAKE (Out_Of_Service TRUE)
7. VERIFY Out_Of_Service = TRUE
8. VERIFY Status_Flags = (?, ?, ?, TRUE)
9. WRITE Present_Value = (PV: (Stages[S].Limit + Stages[S].Deadband) < PV < Stages[S+1].Limit)
10. VERIFY Present_Value = PV
11. VERIFY Present_Stage = S+1
12. VERIFY O1, Present_Value = V1
13. IF (the IUT supports Reliability values other than NO_FAULT_DETECTED) THEN
    REPEAT X = (all values of the Reliability enumeration appropriate to the object type except NO_FAULT_DETECTED *and CONFIGURATION_ERROR*) DO {
    WRITE Reliability = X
    VERIFY Reliability = X
    VERIFY Status_Flags = (?, TRUE, ?, TRUE)
    WRITE Reliability = NO_FAULT_DETECTED
    VERIFY Reliability = NO_FAULT_DETECTED
    VERIFY Status_Flags = (?, FALSE, ?, TRUE)
    }
14. IF (Out_Of_Service is writable) THEN
    WRITE Out_Of_Service = FALSE
    ELSE
    MAKE (Out_Of_Service FALSE)
15. VERIFY Status_Flags = SF1
16. VERIFY Reliability = NO_FAULT_DETECTED

© 2026 by BACnet International. All rights reserved.

17. IF (Present_Stage = S+1) THEN
        VERIFY O1, Present_Value = V2

8. APPLICATION SERVICE INITIATION TESTS

8.2 ConfirmedCOVNotification Service Initiation Tests

8.2.1 Change of Value Notification for Changes to Present_Value in Objects with a COV_Increment

Reason for Change: Defined value for ReportedPV in step 4 for later use.

Purpose: To verify that the IUT can initiate ConfirmedCOVNotification service requests conveying a change of the Present_Value property in Numeric Objects.

Test Concept: A subscription for COV notifications is established, using a Lifetime of L. L shall be set to a value less than 24 hours and large enough to complete the test. The Present_Value of the monitored object is changed by an amount less than the COV increment and it is verified that no COV notification is received. The Present_Value is then changed by an amount greater than the COV increment and a notification shall be received. The Present_Value may be changed using the WriteProperty service or by another means such as changing the input signal represented by an Analog Input object. For some implementations it may be necessary to write to the Out_Of_Service property first to accomplish this task. For implementations where it is not possible to write to these properties at all the vendor shall provide an alternative trigger mechanism to accomplish this task. All of these methods are equally acceptable.

Configuration Requirements: At the beginning of the test, the Out_Of_Service property shall have a value of FALSE. Select an object where Present_Value is not expected to change outside the tester's control by more than COV_Increment or which has a writable Out_Of_Service. In devices where the COV_Increment is always less than the minimal change that Present_Value can make, skip steps 8 through 10.

Notes to Tester: The IUT may initiate additional COVNotifications. The final COVNotification shall accurately reflect Present_Value and Status_Flags.

Test Steps:

REPEAT X = (one supported object of each type) DO {

1. 1. TRANSMIT SubscribeCOV-Request,
   'Subscriber Process Identifier' = (any value > 0 chosen by the TD),
   'Monitored Object Identifier' = X,
   'Issue Confirmed Notifications' = TRUE,
   'Lifetime' = L
2. RECEIVE BACnet-SimpleACK-PDU
3. BEFORE **Notification Fail Time**
4. RECEIVE ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = (the same value used in step 1),
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = X,
   'Time Remaining' = (any value appropriate for the Lifetime selected),
   'List of Values' = (*ReportedPV* = the initial Present_Value and initial Status_Flags)

~~4~~5. TRANSMIT BACnet-SimpleACK-PDU

~~5~~6. TRANSMIT ReadProperty-Request,
   'Object Identifier' = X,
   'Property Identifier' = COV_Increment

~~6~~7. RECEIVE BACnet-ComplexACK-PDU,
   'Object Identifier' = X,
   'Property Identifier' = COV_Increment,
   'Property Value' = (a value "increment" that will be used below)

© 2026 by BACnet International. All rights reserved.

~~7~~*8*. IF (Out_Of_Service is writable) THEN
9. WRITE X, Out_Of_Service = TRUE
10. BEFORE **Notification Fail Time**
11. RECEIVE ConfirmedCOVNotification-Request,
'Subscriber Process Identifier' = (the same value used in step 1),
'Initiating Device Identifier' = IUT,
'Monitored Object Identifier' = X,
'Time Remaining' = (any value appropriate for the Lifetime selected),
'List of Values' = (ReportedPV = the current Present_Value, and new Status_Flags)
12. TRANSMIT BACnet-SimpleACK-PDU
~~8~~*13*.IF (Present_Value is now writable) THEN
14. WRITE X, Present_Value = (any value that differs from ReportedPV by less than "increment")
ELSE
15. MAKE (Present_Value = any value that differs from ReportedPV by less than "increment")
~~9~~*16*.WAIT **Notification Fail Time**
~~10~~*17*. CHECK (verify that no COV notification was transmitted)
~~11~~*18*. IF (Present_Value is now writable) THEN
19. WRITE X, Present_Value = (any value that differs from ReportedPV by an amount greater than "increment")
ELSE
20. MAKE (Present_Value = any value that differs from ReportedPV by an amount greater than "increment")
~~12~~*21*. BEFORE NotificationFailTime
22. RECEIVE ConfirmedCOVNotification-Request,
'Subscriber Process Identifier' = (the same value used in step 1),
'Initiating Device Identifier' = IUT,
'Monitored Object Identifier' = X,
'Time Remaining' = (any value appropriate for the Lifetime selected),
'List of Values' = (the new Present_Value and new Status_Flags)
~~13~~*23*. TRANSMIT BACnet-SimpleACK-PDU
~~14~~*24*. TRANSMIT SubscribeCOV-Request,
'Subscriber Process Identifier' = (the same value used in step 1),
'Monitored Object Identifier' = X
~~15~~*25*. RECEIVE BACnet-SimpleACK-PDU
~~16~~*26*. IF (Out_Of_Service is writable) THEN
27. WRITE X, Out_Of_Service = FALSE
}

8.2.19 Change of Value Notification from Other Standard Object Types (ConfirmedCOVNotification)

Reason for Change: Changed test name for clarity.

Purpose: To verify that the IUT can initiate ConfirmedCOVNotification service requests conveying a change of value for an object (O1) not listed in 135-2020 Table 13-1.

Test Concept: A subscription for COV notifications is established, using a Lifetime of L. L shall be set to a value less than 24 hours and large enough to complete the test. The value of the Present_Value is changed, and it is verified that a COV notification is received. If the Status_Flags is present and can be made to change, it is verified that when Status_Flags changes a COV notification is received.

Configuration Requirements: None.

Test Steps:

1. TRANSMIT SubscribeCOV-Request,
'Subscriber Process Identifier' = (P1, any value □ 0 chosen by the TD),
'Monitored Object Identifier' = O1,

© 2026 by BACnet International. All rights reserved.

'Issue Confirmed Notifications' = TRUE,
'Lifetime' = L
2. RECEIVE BACnet-SimpleACK-PDU
3. BEFORE Notification Fail Time
4. RECEIVE ConfirmedCOVNotification-Request,
'Subscriber Process Identifier' = (P1),
'Initiating Device Identifier' = IUT,
'Monitored Object Identifier' = O1,
'Time Remaining' = (any value appropriate for the Lifetime selected),
'List of Values' = (the initial values for Present_Value and Status_Flags (if O1 supports Status_Flags))
5. TRANSMIT BACnet-SimpleACK-PDU
6. MAKE (Present_Value change)
7. BEFORE Notification Fail Time
8. RECEIVE ConfirmedCOVNotification-Request,
'Subscriber Process Identifier' = (P1),
'Initiating Device Identifier' = IUT,
'Monitored Object Identifier' = O1,
'Time Remaining' = (any value appropriate for the Lifetime selected),
'List of Values' = (updated values for Present_Value and Status_Flags (if O1 supports Status_Flags))
9. IF (Status_Flags are present and can be changed by some action) THEN {
10. MAKE (Status_Flags change)
11. BEFORE Notification Fail Time
12. RECEIVE ConfirmedCOVNotification-Request,
'Subscriber Process Identifier' = (P1),
'Initiating Device Identifier' = IUT,
'Monitored Object Identifier' = O1,
'Time Remaining' = (any value appropriate for the Lifetime selected),
'List of Values' = (updated values for Present_Value, Status_Flags)
13. TRANSMIT BACnet-SimpleACK-PDU
}

8.3 UnconfirmedCOVNotification Service Initiation Tests

8.3.10 Device Restart Notifications

Reason for Change: Clarify the Restart_Notification_Recipients test and the resolution of Time_Of_Device_Restart.

Purpose: To verify that the IUT initiates UnconfirmedCOVNotification service requests to each entry in its Restart_Notification_Recipients property when it resets.

Test Concept: The IUT is configured to send restart notifications and is then reset. The TD checks for the restart notifications.

Device restart notifications differ from subscribed COV notifications that use the UnconfirmedCOVNotification service in two respects. First, subscription is made through the Restart_Notification_Recipients property instead of SubscribeCOV. Second, the 'Subscriber Process Identifier' parameter always has a value of zero.

Configuration Requirements: For each Recipient of the Restart_Notification_Recipients property in the IUT which is of the device form, there shall be a device on the network that will answer Who-Is requests so that the IUT can determine addressing information before sending the restart notification.

Notes to tester: Not all IUTs can accurately differentiate between the types of restart reasons and thus no requirements are placed on the value returned in the restart notification(s). The test shall pass regardless of the order in which the restart notifications are sent to the recipients. If the Restart_Notification_Recipients list has multiple recipients, then the Time_Of_Device_Restart value is expected to be the same in all notifications resulting from the same restart. *Time_Of_Device_Restart must be a specified date and time, however the value is a local matter if time is unknown on restart.*

© 2026 by BACnet International. All rights reserved.

Test Steps:

1. IF (Restart_Notification_Recipients is writable) THEN
2. WRITE(Restart_Notification_Recipients = any non-empty list of Recipients)
ELSE
~~MAKE (Restart_Notification_Recipients contain any non-empty list of Recipients)~~
3. *VERIFY (Restart_Notification_Recipients = local network broadcast)*
~~2~~4. MAKE(the IUT reset)
~~3~~5. REPEAT X = (each entry in the Restart_Notification_Recipients) DO {
6. BEFORE **Notification Fail Time**
7. RECEIVE UnconfirmedCOVNotification-Request,
DESTINATION = X,
'Subscriber Process Identifier' = 0,
'Initiating Device Identifier' = IUT,
'Monitored Object Identifier' = (the IUT Device Identifier),
'Time Remaining' = 0,
'List of Values' = (System_Status=OPERATIONAL,
Time_Of_Device_Restart = (T~~2~~),
Last_Restart_Reason=(any valid restart reason, R))
}
~~4~~8. VERIFY Time_Of_Device_Restart = T~~2~~
~~5~~9. VERIFY Last_Restart_Reason = R
10. *VERIFY Local_Time ~= T.time*
11. *VERIFY Local_Date = T.date*
12. *VERIFY System_Status = OPERATIONAL*
~~6. IF (T2 is not a sequence number) THEN~~
~~VERIFY Local_Time ~= T2~~
~~ELSE~~
~~MAKE(the IUT reset)~~
~~REPEAT X = (each entry in the Restart_Notification_Recipients) DO {~~
~~BEFORE **Notification Fail Time**~~
~~RECEIVE UnconfirmedCOVNotification-Request,~~
~~DESTINATION = X,~~
~~'Subscriber Process Identifier' = 0,~~
~~'Initiating Device Identifier' = IUT,~~
~~'Monitored Object Identifier' = (the IUT Device Identifier),~~
~~'Time Remaining' = 0,~~
~~'List of Values' = (System_Status=OPERATIONAL,~~
~~Time_Of_Device_Restart = (T3),~~
~~Last_Restart_Reason=(any valid restart reason, R))~~
~~7. CHECK (T3 > T2)~~

8.3.21 Change of Value Notification from Other Standard Object Types (UnconfirmedCOVNotification)

Reason for Change: Changed test name for clarity.

Purpose: To verify that the IUT can initiate UnconfirmedCOVNotification service requests conveying a change of value for an object (O1) not listed in 135-2020 Table 13-1.

Test Steps: The steps for this test case are identical to the test steps in 8.2.X2 except that the SubscribeCOV service request in step 1 shall have a value of FALSE for the 'Issue Confirmed Notifications' parameter, all of the ConfirmedCOVNotification requests shall be UnconfirmedCOVNotification requests, and there is no acknowledgment of the unconfirmed services. The MAC address used for the notification message shall be such that the TD is one of the recipients.

© 2026 by BACnet International. All rights reserved.

8.4 ConfirmedEventNotification Service Initiation Tests

8.4.2 CHANGE_OF_STATE Tests (ConfirmedEventNotification)

Reason for change: Optional Parameter “pTimeDelayNormal” has not been included in the Test – Step 18. Renumbered all steps.

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 Multi-state 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.

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 {
3. IF (pMonitoredValue is writable) THEN
4. WRITE pMonitoredValue = (a value from pAlarmValues)
   ELSE
5. MAKE (pMonitoredValue have a value pAlarmValues)
6. WAIT (pTimeDelay)
7. BEFORE **Notification Fail Time**
8. 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,

© 2026 by BACnet International. All rights reserved.

'Message Text' = (optional, any valid message text),
'Notify Type' = EVENT | ALARM,
'AckRequired' = TRUE | FALSE,
'From State' = NORMAL,
'To State' = OFFNORMAL,
'Event Values' = pMonitoredValue, pStatusFlags

9. TRANSMIT BACnet-SimpleACK-PDU
10. IF (Protocol_Revision is present AND Protocol_Revision >= 13) THEN
11. VERIFY Status_Flags = (TRUE, FALSE, ?, ?)
12. VERIFY pCurrentState = OFFNORMAL
13. IF (Protocol_Revision is present AND Protocol_Revision >= 1) THEN
14. VERIFY Event_Time_Stamps = (T1, Ta, Tb)
15. IF (pMonitoredValue is writable) THEN
16. WRITE pMonitoredValue = (a value that corresponds to a NORMAL state)
ELSE
17. MAKE (pMonitoredValue have a value that corresponds to a NORMAL state)
18. WAIT ~~(pTimeDelay)~~ *(pTimeDelayNormal)*
19. BEFORE **Notification Fail Time**
20. 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
21. TRANSMIT BACnet-SimpleACK-PDU
22. IF (Protocol_Revision is present AND Protocol_Revision >= 13) THEN
23. VERIFY Status_Flags = (FALSE, FALSE, ?, ?)
24. VERIFY pCurrentState = NORMAL
25. IF (Protocol_Revision is present AND Protocol_Revision >= 1) THEN
26. VERIFY Event_Time_Stamps = (T1, Ta, T2)
}
~~3~~27. 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 {
28. IF (pMonitoredValue is writable) THEN
29. WRITE pMonitoredValue = (a value from pFaultValues)
ELSE
30. MAKE (pMonitoredValue have a value from pFaultValues)
31. BEFORE **Notification Fail Time**
32. 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,

© 2026 by BACnet International. All rights reserved.

'Message Text' = (optional, any valid message text),
'Notify Type' = EVENT | ALARM,
'AckRequired' = TRUE | FALSE,
'From State' = NORMAL,
'To State' = FAULT,
'Event Values' = pMonitoredValue, pStatusFlags
33. TRANSMIT BACnet-SimpleACK-PDU
34. VERIFY pCurrentState = FAULT
35. IF (Protocol_Revision is present AND Protocol_Revision >= 1) THEN
36. VERIFY Event_Time_Stamps = (Toffnormal, Tfault, Tnormal)
37. VERIFY pCurrentReliability = MULTI_STATE_FAULT
38. IF (pMonitoredValue is writable) THEN
39. WRITE pMonitoredValue = (a value that corresponds to a NORMAL state)
ELSE
40. MAKE (pMonitoredValue have a value that corresponds to a NORMAL state)
41. BEFORE **Notification Fail Time**
42. 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
43. TRANSMIT BACnet-SimpleACK-PDU
44. VERIFY pCurrentState = NORMAL
45. IF (Protocol_Revision is present AND Protocol_Revision >= 1) THEN
46. 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.

8.4.8 CHANGE_OF_LIFE_SAFETY Tests (ConfirmedEventNotification)

8.4.8.7 Mode Transition Tests when Event State is Maintained

Reason for Change: Removed time delay for Mode.

Purpose: To verify the correct operation of the CHANGE_OF_LIFE_SAFETY event algorithm for objects transitioning between the NORMAL, OFFNORMAL and LIFE_SAFETY_ALARM event states when a mode change occurs. Tests are conducted when a mode change occurs, but the event state does not change. Tests are also conducted when a mode change occurs simultaneously with an event state change. In this latter case, the test verifies that the notification is immediate rather than waiting for the time delay.

Test Concept: The object begins the test in a NORMAL state. The Mode is changed. ~~After the time delay expires, the~~ *The* object should transmit an event notification message *immediately*. This operation is tested in the OFFNORMAL and LIFE_SAFETY_ALARM states as well.

~~The test is then repeated by changing the Mode property and simultaneously selecting a pMonitoredValue designated in pAlarmValues. The object should immediately enter the OFFNORMAL state and transmit an event notification message.~~

© 2026 by BACnet International. All rights reserved.

~~pMonitoredValue is then changed to a value corresponding to a NORMAL state, and the Mode is simultaneously written. The object should immediately enter the NORMAL state and transmit an event notification message.~~

Configuration Requirements: The IUT shall be configured such that the Event_Enable property has a value of TRUE for the TO-OFFNORMAL, 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.

Test Steps:

1. VERIFY Event_Detection_Enable = TRUE
2. *VERIFY pCurrentState = NORMAL*
~~2. CHECK (pCurrentState = NORMAL)~~
~~3. MAKE (pMonitoredValue have a value that corresponds to a NORMAL state)~~
4*3*. IF (IUT supports another pMode value which maintains the NORMAL state) THEN {
4. *WRITE pMode = (any valid value not equal to pMode)*
~~MAKE (pMode = different value that maintains pCurrentState as NORMAL)~~
~~WAIT (pTimeDelayNormal)~~
5. BEFORE **Notification Fail Time**
6. RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (any valid process ID),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the intrinsic reporting object 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-NORMAL transition),
'Event Type' = CHANGE_OF_LIFE_SAFETY,
'Message Text' = (S1: optional, any valid message text),
'Notify Type' = EVENT | ALARM,
'AckRequired' = TRUE | FALSE,
'From State' = NORMAL,
'To State' = NORMAL,
'Event Values' = pMonitoredValue, pMode, pStatusFlags, pOperationExpected
7. TRANSMIT BACnet-SimpleACK-PDU
8. IF (Protocol_Revision is present AND Protocol_Revision >= 13) THEN
9. VERIFY Status_Flags = (FALSE, FALSE, ?, ?)
10. VERIFY pCurrentState = NORMAL
~~IF (Protocol_Revision is present AND Protocol_Revision >= 1) THEN~~
11. VERIFY Event_Time_Stamps = (*, *, T1)
12. IF (Event_Message_Texts property exists) THEN
13. VERIFY Event_Message_Texts = (*, *, S1)
}
~~5~~*14*.MAKE (pMonitoredValue have a value that corresponds to an OFFNORMAL state)
15. WAIT (pTimeDelay)
*16. BEFORE **Notification Fail Time***
17. *RECEIVE ConfirmedEventNotification-Request,*
'Process Identifier' = (any valid process ID),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the intrinsic reporting object 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-OFFNORMAL transition),
'Event Type' = CHANGE_OF_LIFE_SAFETY,
'Message Text' = (S2, optional, any valid message text),
'Notify Type' = EVENT | ALARM,
'AckRequired' = TRUE | FALSE,
'From State' = NORMAL,

© 2026 by BACnet International. All rights reserved.

'To State' = OFFNORMAL,
'Event Values' = pMonitoredValue, pMode, pStatusFlags, pOperationExpected
18. TRANSMIT BACnet-SimpleACK-PDU
19. IF (Protocol_Revision is present AND Protocol_Revision >= 13) THEN
20. VERIFY Status_Flags = (TRUE, FALSE, ?, ?)
~~6~~*21.* VERIFY pCurrentState = OFFNORMAL
*22. VERIFY Event_Time_Stamps = (T2, *, T1)*
23. IF (Event_Message_Texts property exists) THEN
*24. VERIFY Event_Message_Texts = (S2, *, S1)*
~~7~~*25.* IF (IUT supports another pMode value which maintains the OFFNORMAL state) THEN {
26. WRITE pMode = (any valid value not equal to pMode)
~~MAKE (pMode = different value that maintains pCurrentState as OFFNORMAL)~~
~~WAIT (pTimeDelay)~~
27. BEFORE **Notification Fail Time**
28. RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (any valid process ID),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the intrinsic reporting object or the Event Enrollment object being tested),
'Time Stamp' = (*T3*~~T2~~: 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_LIFE_SAFETY,
'Message Text' = (*S3*~~S2~~: optional, any valid message text),
'Notify Type' = EVENT | ALARM,
'AckRequired' = TRUE | FALSE,
'From State' = OFFNORMAL,
'To State' = OFFNORMAL,
'Event Values' = pMonitoredValue, pMode, pStatusFlags, pOperationExpected
29. TRANSMIT BACnet-SimpleACK-PDU
30. IF (Protocol_Revision is present AND Protocol_Revision >= 13) THEN
31. VERIFY Status_Flags = (TRUE, FALSE, ?, ?)
32. VERIFY pCurrentState = OFFNORMAL
~~IF (Protocol_Revision is present AND Protocol_Revision >= 1) THEN~~
33. VERIFY Event_Time_Stamps = (*T3*~~T2~~, *, *T1*~~*~~)
34. IF (Event_Message_Texts property exists) THEN
35. VERIFY Event_Message_Texts = (*S3*~~S2~~, *, *S1*~~*~~)
}
~~8~~*36.* MAKE (pMonitoredValue have a value that corresponds to a LIFE_SAFETY_ALARM state)
37. WAIT (pTimeDelay)
*38. BEFORE **Notification Fail Time***
39. RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (any valid process ID),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the intrinsic reporting object or the Event Enrollment object being tested),
'Time Stamp' = (T4: 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_LIFE_SAFETY,
'Message Text' = (S4, optional, any valid message text),
'Notify Type' = EVENT | ALARM,
'AckRequired' = TRUE | FALSE,
'From State' = OFFNORMAL,
'To State' = LIFE_SAFETY_ALARM,
'Event Values' = pMonitoredValue, pMode, pStatusFlags, pOperationExpected
40. TRANSMIT BACnet-SimpleACK-PDU
41. IF (Protocol_Revision is present AND Protocol_Revision >= 13) THEN

© 2026 by BACnet International. All rights reserved.

42. VERIFY Status_Flags = (TRUE, FALSE, ?, ?)
43. VERIFY pCurrentState = OFFNORMAL
*44. VERIFY Event_Time_Stamps = (T4, *, T1)*
45. IF (Event_Message_Texts property exists) THEN
*46. VERIFY Event_Message_Texts = (S4, *, S1)*
~~9~~*47.*IF (IUT supports another pMode value which maintains the LIFE_SAFETY_ALARM state) THEN {
48. WRITE pMode = (any valid value not equal to pMode)
~~MAKE (pMode = different value that maintains pCurrentState = LIFE_SAFETY_ALARM)~~
~~WAIT (pTimeDelay)~~
49. BEFORE **Notification Fail Time**
50. RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (any valid process ID),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the intrinsic reporting object or the Event Enrollment object being tested),
'Time Stamp' = (*T5*~~T3~~: any valid time stamp),
'Notification Class' = (the configured notification class),
'Priority' = (the value configured to correspond to a *TO-OFFNORMAL* ~~TO-NORMAL~~ transition),
'Event Type' = CHANGE_OF_LIFE_SAFETY,
'Message Text' = (*S5*~~S3~~: optional, any valid message text),
'Notify Type' = EVENT | ALARM,
'AckRequired' = TRUE | FALSE,
'From State' = LIFE_SAFETY_ALARM,
'To State' = LIFE_SAFETY_ALARM,
~~'From State' = OFFNORMAL,~~
~~'To State' = OFFNORMAL,~~
'Event Values' = pMonitoredValue, pMode, pStatusFlags, pOperationExpected
51. TRANSMIT BACnet-SimpleACK-PDU
52. IF (Protocol_Revision is present AND Protocol_Revision >= 13) THEN
53. VERIFY Status_Flags = (TRUE, FALSE, ?, ?)
54. VERIFY pCurrentState = OFFNORMAL
~~IF (Protocol_Revision is present AND Protocol_Revision >= 1) THEN~~
55. VERIFY Event_Time_Stamps = (*T5*~~T3~~, *, *T1*~~*~~)
56. IF (Event_Message_Texts property exists) THEN
57. VERIFY Event_Message_Texts = (*S5*~~S3~~, *, *S1*~~*~~)
}

8.4.14 UNSIGNED_RANGE Test (ConfirmedEventNotification Test)

[Note: Test renumbered without using standard change formatting]

Reason for change: wrong property name for high and low limit in Step 23 (Low_Limit and High_Limit instead of pLowLimit and pHighLimit.

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

Test Concept: This test is the same as 8.4.6, except that the Event_Type is UNSIGNED_RANGE instead of OUT_OF_RANGE, and there is no pDeadband. If pMonitoredValue is not under the tester's control in the IUT, then pHighLimit and/or pLowLimit are modified to generate event notifications. The object begins the test in a NORMAL state. pMonitoredValue is raised to a value that is above the high limit. After the time delay expires, the object should enter the HIGH_LIMIT state and transmit an event notification message. pMonitoredValue is lowered to a value that is below the high limit. After the time delay expires, the object should enter the NORMAL state and issue an event notification. The same process is repeated to test the low limit.

Configuration Requirements: If possible, the IUT shall be configured such that the Event_Enable property has a value of TRUE for the TO_OFFNORMAL and TO_NORMAL transitions. If possible, pLimitEnable shall have a value of TRUE for both HighLimit and LowLimit events. The 'Issue Confirmed Notifications' parameter in the Recipient_List of the configured

© 2026 by BACnet International. All rights reserved.

Notification Class shall have a value of TRUE. The Recipient_List of the configured Notification Class shall contain the TD, thus ensuring that notifications are emitted. The event-generating objects shall be in a NORMAL state at the start of the test.

Notes to Tester: The time stamps indicated by "" can have a value that indicates an unspecified time or a time that precedes the timestamp of the first received notification.*

Test Steps:

1. VERIFY pCurrentState = NORMAL
2. IF (pMonitoredValue is writable) THEN
3. WRITE pMonitoredValue = (a value x: (x > pHighLimit))
   ELSE
4. MAKE (pMonitoredValue have a value x: (x > pHighLimit))
~~3~~5. WAIT (pTimeDelay)
~~4~~6. BEFORE **Notification Fail Time**
7. RECEIVE ConfirmedEventNotification-Request,
   'Process Identifier' = (any valid process ID),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (the object being tested),
   'Time Stamp' = (Toffnormal: any valid time stamp),
   'Notification Class' = (the configured notification class),
   'Priority' = (the value configured to correspond to a TO_OFFNORMAL transition),
   'Event Type' = UNSIGNED_RANGE,
   'Message Text' = (optional, any valid message text),
   'Notify Type' = EVENT | ALARM,
   'AckRequired' = TRUE | FALSE,
   'From State' = NORMAL,
   'To State' = HIGH_LIMIT,
   'Event Values' = pMonitoredValue, pStatusFlags, pHighLimit
~~5~~8. TRANSMIT BACnet-SimpleACK-PDU
~~6~~9. IF (Protocol_Revision is present AND Protocol_Revision >= 13)) THEN
10. VERIFY Status_Flags = (TRUE, FALSE, ?, ?)
~~7~~11. VERIFY pCurrentState = HIGH_LIMIT
~~8~~12. IF (Protocol_Revision is present AND Protocol_Revision >= 1) THEN
13. VERIFY Event_Time_Stamps = (Toffnormal, *, *)
~~9~~14. IF (pMonitoredValue is writable) THEN
15. WRITE pMonitoredValue = (a value x: (pLowLimit < x < pHighLimit))
    ELSE
16. MAKE (pMonitoredValue have a value x: (pLowLimit < x < pHighLimit))
~~10~~17. WAIT (pTimeDelayNormal)
~~11~~18. BEFORE **Notification Fail Time**
19. RECEIVE ConfirmedEventNotification-Request,
   'Process Identifier' = (any valid process ID),
   'Initiating Device Identifier' = IUT,
   'Event Object Identifier' = (the object being tested),
   'Time Stamp' = (Tnormal: any valid time stamp),
   'Notification Class' = (the configured notification class),
   'Priority' = (the value configured to correspond to a TO_NORMAL transition),
   'Event Type' = UNSIGNED_RANGE,
   'Message Text' = (optional, any valid message text),
   'Notify Type' = EVENT | ALARM,
   'AckRequired' = TRUE | FALSE,
   'From State' = HIGH_LIMIT,
   'To State' = NORMAL,
   'Event Values' = pMonitoredValue, pStatusFlags, pHighLimit
~~12~~20. TRANSMIT BACnet-SimpleACK-PDU

© 2026 by BACnet International. All rights reserved.

~~13~~*21*. IF (Protocol_Revision is present AND Protocol_Revision >= 13)) THEN
22. VERIFY Status_Flags = (FALSE, FALSE, ?, ?)
~~14~~*23*. VERIFY pCurrentState = NORMAL
~~15~~*24*. IF (Protocol_Revision is present AND Protocol_Revision >= 1) THEN
25. VERIFY Event_Time_Stamps = (Toffnormal, *, Tnormal)
~~16~~*26*. IF (pMonitoredValue is writable) THEN
27. WRITE pMonitoredValue = (a value x: (x < pLowLimit))
ELSE
28. MAKE (pMonitoredValue have a value x: (x < pLowLimit))
~~17~~*29*. WAIT (pTimeDelay)
~~18~~*30*. BEFORE **Notification Fail Time**
31. RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (any valid process ID),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object being tested),
'Time Stamp' = (Tlowlimit: any valid time stamp),
'Notification Class' = (the configured notification class),
'Priority' = (the value configured to correspond to a TO_OFFNORMAL transition),
'Event Type' = UNSIGNED_RANGE,
'Message Text' = (optional, any valid message text),
'Notify Type' = EVENT | ALARM,
'AckRequired' = TRUE | FALSE,
'From State' = NORMAL,
'To State' = LOW_LIMIT,
'Event Values' = pMonitoredValue, pStatusFlags, pLowLimit
~~19~~*32*. TRANSMIT BACnet-SimpleACK-PDU
~~20~~*33*. IF (Protocol_Revision is present AND Protocol_Revision >= 13)) THEN
34. VERIFY Status_Flags = (TRUE, FALSE, ?, ?)
~~21~~*35*. VERIFY pCurrentState = LOW_LIMIT
~~22~~*36*. IF (Protocol_Revision is present AND Protocol_Revision >= 1) THEN
37. VERIFY Event_Time_Stamps = (Tlowlimit, *, Tnormal)
~~23~~*38*. IF (pMonitoredValue is writable) THEN
39. WRITE pMonitoredValue = (a value x: ~~(Low_Limit < x < High_Limit)~~ *(pLowLimit < x < pHighLimit)*)
ELSE
40. MAKE (pMonitoredValue have a value x: ~~(Low_Limit < x < High_Limit)~~ *(pLowLimit < x < pHighLimit)*)
~~24~~*41*. WAIT (pTimeDelayNormal)
~~25~~*42*. BEFORE **Notification Fail Time**
43. RECEIVE ConfirmedEventNotification-Request,
'Process Identifier' = (any valid process ID),
'Initiating Device Identifier' = IUT,
'Event Object Identifier' = (the object being tested),
'Time Stamp' = (Tlowtonormal: any valid time stamp),
'Notification Class' = (the configured notification class),
'Priority' = (the value configured to correspond to a TO_NORMAL transition),
'Event Type' = UNSIGNED_RANGE,
'Message Text' = (optional, any valid message text),
'Notify Type' = EVENT | ALARM,
'AckRequired' = TRUE | FALSE,
'From State' = LOW_LIMIT,
'To State' = NORMAL,
'Event Values' = pMonitoredValue, pStatusFlags, pLowLimit
~~26~~*44*. TRANSMIT BACnet-SimpleACK-PDU
~~27~~*45*. IF (Protocol_Revision is present AND Protocol_Revision >= 13)) THEN
46. VERIFY Status_Flags = (FALSE, FALSE, ?, ?)
~~28~~*47*. VERIFY pCurrentState = NORMAL
~~29~~*48*. IF (Protocol_Revision is present AND Protocol_Revision >= 1) THEN

© 2026 by BACnet International. All rights reserved.

49. VERIFY Event_Time_Stamps = (Tlowlimit, *, Tlowtonormal)

~~Notes to Tester: The time stamps indicated by "*" can have a value that indicates an unspecified time or a time that~~

8.7 GetEnrollmentSummary Service Initiation Tests

[Remove 135.1-2025, Clause 8.7 including all tests]

8.21 ReadRange Service Initiation Tests

8.21.9 Presents Log Records

Purpose: To verify that the IUT can initiate one or more ReadRange requests that access and present a tester-specified portion of log records. It is a generic test used to test data presentation requirements.

Test Concept: Run test in Clause 8.21.8 and verify that the data presentation meets the criteria specified by the BIBB being tested.

Notes to Tester: The values presented by the IUT may differ from the values transmitted on the wire due to rounding, truncation, formatting, language, conversion, etc.

~~Notes to Tester: The IUT is not required to display records containing log-status values.~~

8.44 Who-Am-I Service Initiation Tests

8.44.1 Responds to Who-Is With Who-Am-I While in the Unconfigured State

Reason for change: Removed conditional skip.

Purpose: To verify the IUT will send a Who-Am-I while in the unconfigured state when a Who-Is is received.

Test Concept: TD sends a Who-Is using the wildcard instance of 4194303 and the IUT responds with a Who-Am-I Request.

Test Configuration: The IUT's Device object is not configured with an object instance number. The IUT is configured with a MAC address. ~~If the IUT does not support having a MAC address but no configured Device object instance number, 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 = IUT | LOCAL BROADCAST | GLOBAL BROADCAST | REMOTE BROADCAST,
   Who-Is Request,
   'Device Instance Range Low Limit' = 4194303,
   'Device Instance Range High Limit' = 4194303
2. BEFORE Unconfirmed Response Fail Time
   RECEIVE
   DESTINATION = TD | LOCAL BROADCAST | GLOBAL BROADCAST | REMOTE BROADCAST,
   Who-Am-I Request,
   'Vendor Identifier' = (the IUT's Vendor_Identifier),
   'Model Name' = (the IUT's Model_Name),
   'Serial Number' = (the IUT's Serial_Number)

© 2026 by BACnet International. All rights reserved.

8.45 You-Are Service Initiation Tests

8.45.1 Configures Other Device's MAC Address

Reason for Change: DeviceIdentifier incorrectly included in the You-Are request. Removed MS/TP Subordinate support and fixed WAIT times.

Purpose: To verify the IUT can configure another device's MAC address using the You-Are service without relying on the Who-Am-I service.

Test Concept: The IUT configures TD with an appropriate MAC address without TD first sending a Who-Am-I.

Configuration Requirements: TD's Device object is configured with a known Device object instance number and no configured MAC address.

Notes to Tester: The IUT may require the tester to specify all the parameters needed to configure TD with You-Are, using the IUT's software. The destination address used by TD shall be selected such that the IUT will receive the messages.

Test Steps:

1. MAKE (the IUT configure TD)
2. RECEIVE
   DESTINATION = LOCAL BROADCAST | GLOBAL BROADCAST | REMOTE BROADCAST,
   You-Are Request,
   'Vendor Identifier' = (TD's Vendor_Identifier),
   'Model Name' = (TD's Model_Name),
   'Serial Number' = (TD's Serial_Number),
   ~~'Device Identifier' = (TD's Device object),~~
   'Device MAC Address' = (an appropriate MAC address)

~~3. IF (TD is not an MS/TP subordinate node)~~

3. BEFORE Unconfirmed Response Fail Time

4. TRANSMIT
   DESTINATION = IUT | LOCAL BROADCAST | GLOBAL BROADCAST | REMOTE BROADCAST,
   I-Am Request,
   'Device Identifier' = (TD's Device object)
   'Max APDU Length Accepted' = (any valid value),
   'Segmentation Supported' = (any valid value),
   'Vendor Identifier' = (TD's Vendor_Identifier)

8.45.2 Configures Other Device's Object Instance Number and MAC Address

Reason for Change: Removed MS/TP Subordinate support and fixed WAIT times.

Purpose: To verify the IUT can configure *a* ~~another~~ Device's object instance number and MAC address using the You-Are service without relying on the Who-Am-I service.

Test Concept: The IUT configures TD with an appropriate Device object instance number (X) and MAC address without TD first sending a Who-Am-I.

Configuration Requirements: TD is not configured with a Device object instance number and has no configured MAC address.

Notes to Tester: The IUT will require the tester to specify all the parameters needed to configure TD with You-Are, using the IUT's software. The destination address used by TD shall be selected such that the IUT will receive the messages.

Test Steps:

© 2026 by BACnet International. All rights reserved.

1. MAKE (the IUT configure TD)
~~2. BEFORE Internal Processing Fail Time~~
2. RECEIVE
   DESTINATION = LOCAL BROADCAST | GLOBAL BROADCAST | REMOTE BROADCAST,
   You-Are Request,
   'Vendor Identifier' = (TD's Vendor_Identifier),
   'Model Name' = (TD's Model_Name),
   'Serial Number' = (TD's Serial_Number),
   'Device Identifier' = (Device, X),
   'Device MAC Address' = (any valid MAC address)
~~3. IF (TD is not an MS/TP subordinate node)~~
3. BEFORE Unconfirmed Response Fail Time
4. TRANSMIT
   DESTINATION = IUT | 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' = (TD's Vendor_Identifier)

8.45.3 Can Unconfigure Devices

Reason for Change: Fail time is not necessary for this user action

Purpose: To verify the IUT can unconfigure a device using the You-Are service.

Test Concept: The IUT transmits a You-Are with 'Device Identifier' = (Device, 4194303) in order to unconfigure TD.

Test Steps:

1. MAKE (the IUT *to initiate a You-Are request to* unconfigure *the* TD)
~~2. BEFORE Internal Processing Fail Time~~
~~3~~*2*. RECEIVE
   DESTINATION = TD | LOCAL BROADCAST | GLOBAL BROADCAST | REMOTE BROADCAST,
   You-Are Request,
   'Vendor Identifier' = (TD's Vendor_Identifier),
   'Model Name' = (TD's Model_Name),
   'Serial Number' = (TD's Serial_Number),
   'Device Identifier' = (Device, 4194303),
   'Device MAC Address' = (any valid MAC address, or absent)

9. APPLICATION SERVICE EXECUTION TESTS

9.2 ConfirmedCOVNotification Service Execution Tests

9.2.2 Negative ConfirmedCOVNotification Execution Tests

9.2.2.1 Change of Value Notification Arrives after Subscription has Expired

Reason for Change: Added MAKE step to cancel subscription.

Purpose: To verify that an appropriate error is returned if a COV notification arrives after the subscription time period has expired.

© 2026 by BACnet International. All rights reserved.

Configuration Requirements: If the IUT does not support initiation of SubscribeCOV-Request with 'Issue Confirmed Notifications' equal to TRUE, then this test shall be skipped.

Test Steps:
1. RECEIVE SubscribeCOV-Request,
   'Subscriber Process Identifier' = (any valid process identifier, P1),
   'Monitored Object Identifier' = (any object X of a type that supports COV notification),
   'Issue Confirmed Notifications ' = TRUE,
   'Lifetime' = (any valid Lifetime)
2. TRANSMIT BACnet-SimpleACK-PDU
3. TRANSMIT ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = (P1),
   'Initiating Device Identifier' = TD,
   'Monitored Object Identifier' = X,
   'Time Remaining' = (any amount of time greater than 0),
   'List of Values' = (a list of values appropriate to object X)
4. IF (the IUT can cancel the subscription) THEN
5. *MAKE (the IUT cancel the subscription)*
6. RECEIVE SubscribeCOV – Request,
   'Subscriber Process Identifier' = (PI),
   'Monitored Object Identifier' = X
   ELSE
7. MAKE (the IUT stop resubscribing, if it resubscribes automatically)
~~5~~8. WAIT (at least Lifetime, but sufficient to ensure the subscription has expired)
~~6~~9. TRANSMIT ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = (P1),
   'Initiating Device Identifier' = TD,
   'Monitored Object Identifier' = X,
   'Time Remaining' = (any amount of time greater than 0),
   'List of Values' = (a list of values appropriate to object X)
~~7~~10. IF (Protocol_Revision is present and Protocol_Revision >= 10) THEN
11. RECEIVE BACnet-Error-PDU,
   'Error Class' = SERVICES,
   'Error Code' = UNKNOWN_SUBSCRIPTION |
   (BACnet-SimpleACK-PDU)
   ELSE
12. RECEIVE BACnet-Error-PDU,
   'Error Class' = SERVICES,
   'Error Code' = (any valid error code for class SERVICES) |
   (BACnet-SimpleACK-PDU)

9.10 SubscribeCOV Service Execution Tests

9.10.1 Positive SubscribeCOV Service Execution Tests

9.10.1.11 Ensuring 5 Concurrent COV Subscribers

Reason For Change: Clarified the Test Concept and added Test Conditionality. Changed test to support 5 subscriptions from any number of objects.

Purpose: This test case verifies that the IUT can support 5 concurrent subscriptions.

Test Concept: Have the TD subscribe with 5 different process identifiers, *V1 through V5 to monitored objects O1 through O5*~~V1 through V5 for monitored object O1~~. After each subscription*, verify the IUT initiates the* ~~verify a~~ corresponding

© 2026 by BACnet International. All rights reserved.

notification ~~is sent~~. Change *the value of* the monitored object and verify *the IUT initiates the corresponding notification.* ~~that all 5 notifications were sent.~~

Test Conditionality: The IUT should not have any subscriptions at the start of this test.

Notes to Tester: ~~The notification in step 3 can be received in any order by the TD.~~ *Object O1 through O5 can be the same object if 5 COV subscriptions are supported in a single object or different objects if no object supports 5 COV subscriptions.*

Test Steps

1. REPEAT (*x = 1 to 5*~~X=V1 to V5~~) DO {
2. TRANSMIT SubscribeCOV-Request,
   'Subscriber Process Identifier' = *Vx*~~X~~,
   'Monitored Object Identifier' = *Ox*~~O1~~,
   'Issue Confirmed Notifications' = TRUE | FALSE,
   'Lifetime' = (any valid value that will allow the subscription to outlast the test)
3. RECEIVE BACnet-SimpleACK-PDU
4. IF (confirmed notifications ~~were~~ requested) THEN
5. BEFORE **Notification Fail Time**
6. RECEIVE ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = *Vx*~~X~~,
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = *Ox*~~O1~~,
   'Time Remaining' = (any valid value),
   'List of Values' = (the initial Present_Value and initial Status_Flags)
7. TRANSMIT BACnet-SimpleACK-PDU
   ELSE
8. BEFORE **Notification Fail Time**
9. RECEIVE UnconfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = *Vx*~~X~~,
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = *Ox*~~O1~~,
   'Time Remaining' = (any valid value),
   'List of Values' = (the initial Present_Value and initial Status_Flags)
   }

~~10. MAKE (Present_Value = any value that differs from "initial Present_Value" such that a COV notification is generated)~~
10. REPEAT (*x = 1 to 5*~~X=V1 to V5~~) DO {
11. MAKE (Ox, Present_Value change such that the IUT initiates the required COV notifications)
12. IF (~~if~~ confirmed notifications ~~were~~ requested) THEN
13. RECEIVE ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = *Vx*~~X~~,
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = *Ox*~~O1~~,
   'Time Remaining' = (any valid value),
   'List of Values' = (the new Present_Value and Status_Flags)
14. TRANSMIT BACnet-SimpleACK-PDU
   ELSE
15. RECEIVE UnconfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = *Vx*~~X~~,
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = *Ox*~~O1~~,
   'Time Remaining' = (any valid value),
   'List of Values' = (the new Present_Value and Status_Flags)
   }

© 2026 by BACnet International. All rights reserved.

9.13 AtomicWriteFile Service Execution Tests

9.13.1 Positive AtomicWriteFile Service Execution Tests

9.13.1.2 Writing to Stream-Based Files

9.13.1.2.1 Writing an Entire Stream-Based File

Reason for Change: Update in Step 4 and Step 6 considering possibility that File_size can be 0.

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~~
4. *IF (File_Size is not equal to the size of the test file AND File_Size != 0) 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~~ *file size of the test data or a value of 0*)
7. VERIFY Modification_Date = (the current date and time)
8. VERIFY ARCHIVE = FALSE

9.14 AddListElement Service Execution Tests

9.14.2 Negative AddListElement Service Execution Tests

9.14.2.2 Adding a List Element With an Invalid Datatype

Reason for Change: Cleanup reject codes.

Purpose: To verify the ability of the IUT to correctly respond to an AddListElement service request to add an element with an invalid datatype to a list.

© 2026 by BACnet International. All rights reserved.

Notes to Tester: *In Step #1, an 'inappropriate' datatype means a datatype never supported by the property being tested.* ~~value selected for step 1 is 'inappropriate', not a value which is 'allowed', but not supported by this instance of the property, i.e., it is not one of the datatypes that would ever be supported by an instance of this property in this object type.~~ *Also,* DATATYPE_NOT_SUPPPORTED is only *valid* ~~correct~~ when the datatype requested is supported, for example, in a CHOICE, by this property in this object type, but not supported by this instance of the property.

Test Steps:

1. TRANSMIT AddListElement-Request,
   'Object Identifier' = L,
   'Property Identifier' = ListProp,
   'List of Elements' = (a single element with a datatype inappropriate for this property)
2. RECEIVE
   (AddListElement-Error,
   Error Class =PROPERTY,
   Error Code =INVALID_DATATYPE,
   'First Failed Element' = 1) |
   (BACnet-Reject-PDU
~~Reject Reason = INVALID_PARAMETER_DATATYPE) |~~
~~(BACnet-Reject-PDU~~
~~Reject Reason = INVALID_TAG)~~
   Reject Reason = (any reject reason))

9.14.2.3 An AddListElement Failure Part Way Through a List

Reason for Change: Cleanup reject codes.

Purpose: To verify the ability of the IUT to respond to an AddListElement service request to add multiple elements to a list where one of the elements cannot be added. Upon failure, the AddListElement service should leave the list unchanged.

Notes to Tester: The value selected for step 2 is 'inappropriate', not a value which is 'allowed', but not supported by this instance of the property, i.e., it is not one of the datatypes that would ever be supported by an instance of this property in this object type. DATATYPE_NOT_SUPPPORTED is only correct when the datatype requested is supported, for example, in a CHOICE, by this property in this object type, but not supported by this instance of the property. Care must be taken that the inserted error is not likely to cause confusion in the IUT as to which element contains the error.

Test Steps:

1. READ InitialList = (L), ListProp
2. TRANSMIT AddListElement-Request,
   'Object Identifier' = L,
   'Property Identifier' = ListProp
   'List of Elements' = (two or more elements to be added to the list with the second element
   having the wrong datatype)
~~2. IF (Protocol_Revision is present AND Protocol_Revision >= 7) THEN~~
3. RECEIVE
   AddListElement-Error,
   Error Class =PROPERTY,
   Error Code =INVALID_DATATYPE
   'First Failed Element' = 2
   | (~~RECEIVE~~ BACnet-Reject-PDU,
~~Reject Reason = INVALID_TAG | INVALID_PARAMETER_DATA_TYPE)~~
   Reject Reason = (any reject reason))
~~ELSE~~
~~RECEIVE~~
~~AddListElement-Error,~~

© 2026 by BACnet International. All rights reserved.

~~Error Class =SERVICES,~~
~~Error Code =INVALID_PARAMETER_DATATYPE~~
~~'First Failed Element' = 2~~
~~| (AddListElement-Error,~~
~~'Error Clas's = PROPERTY,~~
~~'Error Code' = INVALID_DATATYPE,~~
~~'First Failed Element' = 2)~~
~~| (BACnet-Reject-PDU,~~
~~Reject Reason = INVALID_TAG | INVALID_PARAMETER_DATA_TYPE)~~
~~3~~*4*. VERIFY (L), ListProp = InitialList

9.16 CreateObject Service Execution Tests

9.16.2 Negative CreateObject Service Execution Tests

9.16.2.3 Attempting to Create an Object with an Object Identifier That is Not Creatable by Specifying the Object Identifier

Reason for Change: Clarify the object to be used.

Purpose: To verify the correct execution of the CreateObject service request when the 'Object Specifier' parameter conveys an object identifier for an object type that is not dynamically creatable in the IUT.

Test Steps:

1. TRANSMIT CreateObject-Request,
   'Object Identifier' = (any ~~object identifier having a~~ supported object type for which dynamic creation *using the CreateObject service* is not supported)
2. RECEIVE CreateObject-Error,
   Error Class = OBJECT,
   Error Code = DYNAMIC_CREATION_NOT_SUPPORTED
   'First Failed Element Number' = 0
3. VERIFY (the IUT's Device object),
   Object_List = (any object list that does not contain the object specified in step 1)

~~Notes to tester: If the IUT limits the instances that can be created, this shall be taken into account when selecting an object identifier in step 1.~~

9.16.2.4 Attempting to Create an Object with an Object Type Specifier and an Error in the Initial Values

Reason for Change: Cleanup reject codes.

Purpose: To verify the correct execution of the CreateObject service request when an object type is used as the object specifier and a list of initial property values containing an invalid value is provided.

Test Concept: The TD shall attempt to create an object with an object type specifier and the 'List Of Initial Values' parameter containing a value which is out of range. The TD then attempts to create an object with a value of an inappropriate datatype in the 'List Of Initial Values' parameter. The selected datatype is not compliant with the property definition given by the BACnet standard.

Configuration Requirements: The value to be written shall not be of a datatype which is compliant with the property definition, but which is not supported for P1 by the IUT. For instance, Schedule_Default, which is defined to be of Any primitive datatype, would not be used in this test along with BitString datatype, even where the IUT's Schedule object cannot be configured for scheduling BitString values.

Test Steps:

© 2026 by BACnet International. All rights reserved.

1. READ X1 = Object_List
2. TRANSMIT CreateObject-Request,
   'Object Specifier' = (*O1,* any creatable object type),
   'List Of Initial Values' = (a list of one or more properties and their initial values, that the IUT will accept initial values for, with one of the values being out of range)
~~3. IF (Protocol_Revision is present AND Protocol_Revision >= 4) THEN~~
3. RECEIVE CreateObject-Error PDU,
   Error Class = PROPERTY,
   Error Code = VALUE_OUT_OF_RANGE
   'First Failed Element Number' = (the position in the 'List Of Initial Values' with the offending value)
~~ELSE~~
~~RECEIVE CreateObject-Error,~~
~~Error Class = PROPERTY,~~
~~Error Code = VALUE_OUT_OF_RANGE | OTHER~~
~~'First Failed Element Number' = (the position in the 'List Of Initial Values' with the offending value)~~
4. READ X2 = Object_List
5. VERIFY X1 = X2
~~4. CHECK(Verify that the new object was not created)~~
~~5~~*6*. TRANSMIT CreateObject-Request,
   'Object Specifier' = (*O1*~~object type from step 2~~),
   'List Of Initial Values' = (a list of one or more properties and their initial values, that the IUT will accept initial values for, with one of the values being an inappropriate datatype)
~~6. IF (Protocol_Revision is present AND Protocol_Revision >= 4) THEN~~
7. RECEIVE
   CreateObject-Error,
      Error Class = PROPERTY,
      Error Code = INVALID_DATATYPE
      'First Failed Element Number' = (the position in the 'List Of Initial Values' with the offending value)
   | (BACnet-Reject-PDU
~~Reject Reason = INVALID_PARAMETER_DATATYPE | INVALID_TAG)~~
      Reject Reason = (any reject reason))
~~ELSE~~
~~RECEIVE~~
~~CreateObject-Error,~~
~~Error Class = PROPERTY,~~
~~Error Code = VALUE_OUT_OF_RANGE | INVALID_DATATYPE | OTHER~~
~~'First Failed Element Number' = (the position in the 'List Of Initial Values' with the offending value)~~
~~| (BACnet-Reject-PDU~~
~~Reject Reason = INVALID_PARAMETER_DATATYPE | INVALID_TAG)~~
~~7~~*8*. READ X2 = Object_List
9. VERIFY X1 = X2
~~8. CHECK (X1 = X2)~~

9.16.2.5 Attempting to Create an Object with an Object Identifier and an Error in the Initial Values

Reason for Change: Cleanup reject codes.

Purpose: To verify the correct execution of the CreateObject service request when an object identifier is used as the object specifier and a list of initial property values containing an invalid value is provided.

Test Concept: The TD shall attempt to create an object with an object type specifier and the 'List Of Initial Values' parameter containing a value which is out of range. The TD then attempts to create an object with a value of an inappropriate datatype in the 'List Of Initial Values' parameter. The selected datatype is not compliant with the property definition given by the BACnet standard.

© 2026 by BACnet International. All rights reserved.

Configuration Requirements: The value to be written shall not be of a datatype which is compliant with the property definition, but which is not supported for P1 by the IUT. For instance, Schedule_Default, which is defined to be of Any primitive datatype, would not be used in this test along with BitString datatype, even where the IUT's Schedule object cannot be configured for scheduling BitString values.

Test Steps:

1. TRANSMIT CreateObject-Request,
   'Object Specifier' = (*O1,* any unique object identifier of a type that is creatable and an instance number that is creatable),
   'List Of Initial Values' = (a list of one or more properties and their initial values, that the IUT will accept initial values for, with one of the values *at location L1 in the list,* being out of range)
~~2. IF (Protocol_Revision is present AND Protocol_Revision >= 4) THEN~~
2. RECEIVE CreateObject-Error,
   Error Class = PROPERTY,
   Error Code = VALUE_OUT_OF_RANGE
   'First Failed Element Number' = *L1*~~(the position in the 'List Of Initial Values' with the offending value)~~
~~ELSE~~
~~RECEIVE CreateObject-Error,~~
~~Error Class = PROPERTY,~~
~~Error Code = VALUE_OUT_OF_RANGE | OTHER~~
~~'First Failed Element Number' = (the position in the 'List Of Initial Values' with the offending value)~~
3. READ X2 = Object_List
4. VERIFY X1 = X2
~~3. CHECK (Verify that the new object was not created)~~
4*5*. TRANSMIT CreateObject-Request,
   'Object Specifier' = *O1*~~ (object identifier from step 1)~~,
   'List Of Initial Values' = (a list of one or more properties and their initial values, that the IUT will accept initial values for, with one of the values *at location L1 in the list,* being an inappropriate datatype)
~~5. IF (Protocol_Revision is present AND Protocol_Revision >= 4) THEN~~
6. RECEIVE
   CreateObject-Error,
   Error Class = PROPERTY,
   Error Code = INVALID_DATATYPE
   'First Failed Element Number' = *L1*~~(the position in the 'List Of Initial Values' with the offending value)~~
   | (BACnet-Reject-PDU
~~Reject Reason = INVALID_PARAMETER_DATATYPE) |~~
~~| (BACnet-Reject-PDU~~
~~Reject Reason = INVALID_TAG)~~
Reject Reason = (any reject reason))
~~ELSE~~
~~RECEIVE~~
~~CreateObject-Error,~~
~~Error Class = PROPERTY,~~
~~Error Code = VALUE_OUT_OF_RANGE | INVALID_DATATYPE | OTHER~~
~~'First Failed Element Number' = (the position in the 'List Of Initial Values' with the offending value)~~
~~| (BACnet-Reject-PDU~~
~~Reject Reason = INVALID_PARAMETER_DATATYPE | INVALID_TAG)~~
6*7*. TRANSMIT ReadProperty-Request,
   'Object Identifier' = *O1*~~ (the 'Object Identifier' used in step 1)~~,
   'Property Identifier' = Object_Name
~~7. IF (Protocol_Revision is present AND Protocol_Revision >= 4) THEN~~
8. RECEIVE BACnet-Error PDU,
   Error Class = OBJECT,
   Error Code = UNKNOWN_OBJECT

© 2026 by BACnet International. All rights reserved.

~~ELSE~~
~~RECEIVE BACnet-Error PDU~~
~~Error Class = OBJECT,~~
~~Error Code = UNKNOWN_OBJECT | NO_OBJECTS_OF_SPECIFIED_TYPE | OTHER~~

9.16.2.8 Attempting to Create a non-Supported Object Type (by Object Identifier)

Reason for Change: Clarified the type of object to be used.

Purpose: To verify the correct execution of the CreateObject service request when the 'Object Specifier' parameter conveys an object identifier for an object type that is not supported in the IUT.

~~Notes to Tester: If the IUT limits the instances that can be created, this shall be taken into account when selecting an object identifier in step 1.~~

Test Steps:
1. TRANSMIT CreateObject-Request,
   'Object Specifier' = (any object identifier having an unsupported object type)
2. IF (Protocol_Revision >= 10) THEN
   RECEIVE CreateObject-Error,
   'Error Class' = OBJECT,
   'Error Code' = UNSUPPORTED_OBJECT_TYPE
   'First Failed Element Number' = 0
   ELSE
   RECEIVE CreateObject-Error,
   'Error Class' = (any valid error class),
   'Error Code' = (any valid error code)
   'First Failed Element Number' = 0
3. VERIFY (the IUT's Device object, Object_List = (any object list that does not contain the object specified in step 1)

9.18 ReadProperty Service Execution Tests

9.18.1 Positive ReadProperty Service Execution Tests

9.18.1.6 Respects max-segments-accepted bit pattern

Reason for Change: Update test to include missing parameter in Step 1. Remove ReadRange as one of the suggested services that could be used for the test, since a proper implementation of ReadRange will not result in an abort and would instead send fewer results with the More Items flag set.

Purpose: To verify that the IUT abides by the 'max-segments-accepted' parameter, when the size of the response ~~does require~~*requires* segmentation.

Configuration Requirements: ~~Use a very small 50 octet 'max-APDU-length-accepted' size in the request. The BACnetConfirmed-Request-PDU shall be one where the response size will exceed 2 times 'max-APDU-length-accepted' and so require at least three segments. If the largest response that the IUT can return is 100 or fewer octets, then this test shall be skipped.~~ *The TD shall be configured to issue a BACnet-Confirmed-Request-PDU, specifying a max-apdu-length-accepted of 50-octets, max-segments-accepted equal to 2, and segmented-response-accepted equal to True. The total response from the IUT should exceed three 50-octet segments, triggering the IUT to respond with a BUFFER_OVERFLOW.*

Notes to Tester: ~~An attempt to read the whole Object_List might suffice. Or a ReadRange or ReadPropertyMultiple or AtomicReadFile request, if any of those services are executed.~~*An attempt to read the whole Object_List may satisfy the*

© 2026 by BACnet International. All rights reserved.

requirements of this test. If the IUT supports execution of ReadPropertyMultiple service or AtomicReadFile service, then one of those services may also satisfy the requirements of this test.

Test Steps:

1. TRANSMIT BACnet-Confirmed-Request-PDU,
   'segmented-response-accepted' = TRUE
   'max-segments-accepted' = ~~2~~*'B'001', -- 2 segments accepted*
   'max-apdu-length-accepted' = B'0000', -- minimum message size, 50 octets
2. RECEIVE BACnet-Abort-PDU,
   'Abort Reason' = BUFFER_OVERFLOW

9.18.1.7 Reading Array Properties at Different Array Indexes

Reason for Change: Update to account for changed language in 135-2020ci-6.

Purpose: This test verifies the IUT can execute ReadProperty service requests on a single element in an array property.

Test Concept: This test will execute a ReadProperty service request to read a single element from the selected property by specifying the array-index in the request. Another request is made to read an element of an array where the array index is *greater than the current length of the array.*~~out of range.~~

Configuration Requirement: O1 is any object in the IUT database having array property P1 having size X.

Test Steps:

1. VERIFY P1 = X, ARRAY INDEX = 0
2. IF (X>0) THEN
   READ V = P1, ARRAY INDEX = 1
   CHECK (V is any valid value of the correct data type for property P1)
   READ V = P1, Array Index =X
   CHECK (V is any valid value of the correct data type for property P1)
3. TRANSMIT ReadProperty-Request,
   'Object Identifier' = O1,
   'Property Identifier' = P1
   'Property Array Index' = (X+1)
4. RECEIVE BACnet-Error-PDU,
   'Error Class' = PROPERTY,
   'Error Code' = INVALID_ARRAY_INDEX

9.18.2 Negative ReadProperty Service Execution Tests

9.18.2.2 Reading Array Properties With An Array Index That Is Out Of Range

Reason for Change: Remove this test.

~~Purpose: To verify that the IUT can execute ReadProperty service requests when the requested property value is an array but the ARRAY INDEX is out of range.~~

~~Test Steps:~~

~~1. TRANSMIT ReadProperty-Request,~~
~~'Object Identifier' = (Device, X),~~
~~'Property Identifier' = Object_List,~~
~~'Priority Array Index' = (a value larger than the size of the Object_List)~~
~~2. RECEIVE BACnet-Error-PDU,~~
~~Error Class = PROPERTY,~~
~~Error Code = INVALID_ARRAY_INDEX~~

© 2026 by BACnet International. All rights reserved.

9.20 ReadPropertyMultiple Service Execution Tests

9.20.1 Positive ReadPropertyMultiple Service Execution Tests

9.20.1.7 Reading ALL Properties

Reason for change: Modify 'Notes to Tester' to clarify acceptable proprietary property errors. Remove mentions of PR 7. Add test numbers.

Purpose: To verify the ability to correctly execute a ReadPropertyMultiple service request that uses the special property identifier ALL. One instance of each object-type supported is tested.

Notes to Tester: If a property which is not readable using the ReadPropertyMultiple service is in the specified object, ~~and Protocol_Revision < 7, then either no entry is returned, or an error code is returned. If Protocol_Revision >= 7,~~ then the entry shall contain 'Error Class': PROPERTY and 'Error-Code': READ_ACCESS_DENIED for ~~that property~~ *standard properties, and relevant errors from clause 18.3 of the BACnet Standard for proprietary properties*. Property_List (371) shall not appear in the List of Results.

Test Steps:

1. REPEAT ObjectX = (one instance of each supported object type) DO {
2. TRANSMIT ReadPropertyMultiple-Request,
   'Object Identifier' = ObjectX,
   'Property Identifier' = ALL
3. RECEIVE ReadPropertyMultiple-ACK,
   'Object Identifier' = ObjectX,
   'List Of Results' = (a list of all standard properties
   documented for ObjectX in the EPICS plus all proprietary
   properties present in ObjectX, each with a valid
   value, excluding the Property_List property)

}

9.20.1.16 ReadPropertyMultiple Array Properties

Reason for Change: Correct test issue that occurred when array length was zero.

Purpose: To verify that the IUT can execute ReadPropertyMultiple service requests when the requested property is an array~~, when its size as well as when a single element of the array is requested. Another request is made to read an element of an array where the array index is out of range~~.

Test Concept: The TD reads the size of the array property, and then reads the first and last entries in the array. Finally, the TD reads past the end of the array and ensures that the IUT returns the correct error.

Configuration Requirement: O1 is any object in the IUT database having array property P1.

Test Steps:

1. ~~VERIFY P1 = X,~~*READ X = (O1), P1,* ARRAY INDEX = 0
2. IF (X>0) THEN
3. TRANSMIT ReadPropertyMultiple-Request,
   'Object Identifier' = O1,
   'Property Identifier' = P1,
   'Property Array Index' = 1
4. RECEIVE ReadPropertyMultiple-ACK,
   'Object Identifier' = O1,

© 2026 by BACnet International. All rights reserved.

'Property Identifier' = P1,
'Property Array Index' = 1,
'Property Value' = (V, any valid value of the correct data type for property P1)

5. TRANSMIT ReadPropertyMultiple-Request,
'Object Identifier' = O1,
'Property Identifier' = P1,
'Property Array Index' = X,

6. RECEIVE ReadPropertyMultiple-ACK,
'Object Identifier' = O1,
'Property Identifier' = P1,
'Property Array Index' = X,
'Property Value' = (V, any valid value of the correct data type for property P1)

~~7. CHECK (V is any value of the correct data type for property P1)~~

~~8~~*7*. TRANSMIT ReadPropertyMultiple-Request,
'Object Identifier' = O1,
'Property Identifier' = P1,
'Property Array Index' = (X+1)

~~9~~*8*. RECEIVE ReadPropertyMultiple-Error,
'Error Class' = PROPERTY,
'Error Code' = INVALID_ARRAY_INDEX
| ReadPropertyMultiple-ACK,
'Object Identifier' = O1,
'Property Identifier' = P1,
'Property Array Index' = X+1,
'Property Access Error' = (
'Error Class' = PROPERTY,
'Error Code' = INVALID_ARRAY_INDEX)

9.20.2 Negative ReadPropertyMultiple Service Execution Tests

9.20.2.3 Reading a Single Non-Array Property with an Array Index

Reason for Change: Remove PR4 error codes.

Purpose: This test case verifies that the IUT can execute ReadPropertyMultiple service requests when the requested property value is not an array but an ARRAY INDEX is included in the service request. ~~This test shall only be performed if Protocol_Revision is present and has a value greater than or equal to 4.~~

Test Steps:

1. TRANSMIT ReadPropertyMultiple-Request,
'Object Identifier' = (Device, X),
'Property Identifier' = Vendor_Name,
'Priority Array Index' = 1

~~2. IF (Protocol_Revision is present AND Protocol_Revision >= 4) THEN~~

2. RECEIVE
(BACnet-Error-PDU,
'Error Class' = PROPERTY,
'Error Code' = PROPERTY_IS_NOT_AN_ARRAY) |
(ReadPropertyMultiple-ACK,
'Object Identifier' = (Device, X),
'Property Identifier' = Vendor_Name,
'Priority Array Index' = 1,
'Error Class' = PROPERTY,
'Error Code' = PROPERTY_IS_NOT_AN_ARRAY)

~~ELSE~~
~~RECEIVE~~

© 2026 by BACnet International. All rights reserved.

~~(BACnet-Reject-PDU,~~
~~'Reject Reason' = INCONSISTENT_PARAMETERS) |~~
~~(BACnet-Reject-PDU,~~
~~'Reject Reason' = INVALID_TAG)|~~
~~(BACnet-Error-PDU,~~
~~'Error Class' = PROPERTY,~~
~~'Error Code' = INVALID_ARRAY_INDEX) |~~
~~(ReadPropertyMultiple-ACK,~~
~~'Object Identifier' = (Device, X),~~
~~'Property Identifier' = Vendor_Name,~~
~~'Priority-Array-Index' = 1,~~
~~'Error Class' = PROPERTY,~~
~~'Error Code' = INVALID_ARRAY_INDEX) |~~
~~(ReadPropertyMultiple-ACK,~~
~~'Object Identifier' = (Device, X),~~
~~'Property Identifier' = Vendor_Name,~~
~~'Priority-Array-Index' = 1~~
~~'Error Class' = SERVICES,~~
~~'Error Code' = INCONSISTENT_PARAMETERS) |~~
~~(BACnet-Error-PDU,~~
~~'Error Class' = SERVICES,~~
~~'Error Code' = INCONSISTENT_PARAMETERS) |~~
~~(BACnet-Error-PDU,~~
~~'Error Class' = PROPERTY,~~
~~'Error Code' = PROPERTY_IS_NOT_AN_ARRAY) |~~
~~(ReadPropertyMultiple-ACK,~~
~~'Object Identifier' = (Device, X),~~
~~'Property Identifier' = Vendor_Name,~~
~~'Priority-Array-Index' = 1~~
~~'Error Class' = PROPERTY,~~
~~'Error Code' = PROPERTY_IS_NOT_AN_ARRAY)~~

9.22 WriteProperty Service Execution Tests

9.22.1 Positive WriteProperty Service Execution Tests

9.22.1.6 Writing NULL to Non-commandable Properties

Reason for Change: The standard was changed in PR21 to require that devices not return errors when a Relinquish~~NULL~~ is written to writable *optionally* non-commandable Present_Value properties.

Purpose: This test case verifies that the IUT returns a Result(+) when an attempt is made to *relinquish the Present_Value property, P1, of an instance of an object, O1, that is writable, non-commandable, and is an optionally commandable object type.* ~~relinquish a *writable* non-commandable *Present_Value* property.~~

Test Concept: Write NULL*, at a priority to*~~, to a writable non-commandable *Present_Value* property,~~ P1 in object O1, and verify the IUT returns a Result(+) and does not modify the property.

Test Configuration: *None.*~~P1 shall be a property for which NULL is not an accepted value.~~

Test Steps:
1. READ X = (O1), P1
2. TRANSMIT WriteProperty-Request,
   'Object Identifier' = O1,
   'Property Identifier' = P1,

© 2026 by BACnet International. All rights reserved.

'Property Value' = NULL
'Priority' = (any valid value)
3. RECEIVE BACnet-SimpleACK-PDU
4. VERIFY (O1), P1 = X
5. *TRANSMIT WriteProperty-Request,*
'Object Identifier' = O1,
'Property Identifier' = P1,
'Property Value' = NULL,
6. *RECEIVE BACnet-SimpleACK-PDU*
7. *VERIFY (O1), P1 = X*

9.22.2 Negative WriteProperty Service Execution Tests

9.22.2.1 Writing Non-Array Properties with an Array Index

Reason for Change: Add negative testing when index = 0.

Purpose: To verify that the IUT can execute WriteProperty service requests when the property value is not an array but an array index is included in the service request.

Test Concept: The TD shall select an object in the IUT that contains a writable scalar property designated P1. An attempt will be made to write to this property using an ARRAY INDEX *that is a positive integer and again with and an ARRAY INDEX = 0*. If no suitable object can be found, then this test shall be omitted.

Configuration Requirements: If the IUT supports any writable properties that are scalars, it shall be configured with at least one such property that can be used for this test.

Test Steps:
1. READ X = (Object1), P1
2. TRANSMIT WriteProperty-Request,
'Object Identifier' = Object1,
'Property Identifier' = P1,
'Property Value' = (any valid value of the correct datatype for this property subject to
the restrictions specified in the EPICS as defined in 4.4.2, except
the value X read in step 1),
'Property Array Index' = (any positive integer)
3. IF (Protocol_Revision is present AND Protocol_Revision >= 4) THEN
4. RECEIVE BACnet-Error PDU,
Error Class = PROPERTY,
Error Code = PROPERTY_IS_NOT_AN_ARRAY
ELSE
5. RECEIVE BACnet-Error PDU,
Error Class = SERVICES,
Error Code = INCONSISTENT_PARAMETERS
46. VERIFY (Object1), P1 = X
7. *TRANSMIT WriteProperty-Request,*
'Object Identifier' = Object1,
'Property Identifier' = P1,
'Property Value' = (any valid value of the correct datatype for this property subject to
the restrictions specified in the EPICS as defined in 4.4.2, except
the value X read in step 1),
'Property Array Index' = 0
8. *IF (Protocol_Revision is present AND Protocol_Revision >= 4) THEN*
9. *RECEIVE BACnet-Error PDU,*
Error Class = PROPERTY,
Error Code = PROPERTY_IS_NOT_AN_ARRAY

© 2026 by BACnet International. All rights reserved.

ELSE
10. *RECEIVE BACnet-Error PDU,*
Error Class = SERVICES,
Error Code = INCONSISTENT_PARAMETERS
11. *VERIFY (Object1), P1 = X*

9.22.2.2 Writing Array Properties with an Array Index that is *Greater Than The Length Of The Array*~~Out of Range~~

Reason for Change: Update to account for new language in 135-2020ci-6. Add single quotes around Error Class and Error Code in step 3.

Purpose: To verify that the IUT can execute WriteProperty service requests when the requested property value is an array but the array index is *greater than the length of the array.*~~out of range.~~

Test Concept: The TD shall select an object in the IUT that contains a writable array property designated P1. An attempt will be made to write to this property using an ARRAY INDEX that is *greater than the length of the array.*~~out of range.~~ If no suitable object can be found, then this test shall be omitted.

Configuration Requirements: If the IUT supports any writable properties that are arrays, it shall be configured with at least one such property that can be used for this test.

Test Steps:

1. READ X = (Object1), P1
2. TRANSMIT WriteProperty-Request,
'Object Identifier' = Object1,
'Property Identifier' = P1,
'Property Value' = (any valid value of the correct datatype for this property subject to the restrictions specified in the EPICS as defined in 4.4.2, except the value X ~~read in step 1~~),
'Property Array Index' = (any value that is ~~larger than the current size~~ *greater than the current length* of the array)
3. RECEIVE BACnet-Error PDU,
'Error Class' = PROPERTY,
'Error Code' = INVALID_ARRAY_INDEX
4. VERIFY (Object1), P1 = X

9.22.2.3 Writing with a Property Value Having the Wrong Datatype

Reason for Change: Cleanup reject codes.

Purpose: To verify that the IUT correctly responds to an attempt to write a property value that has an invalid datatype.

Test Concept: The TD shall select an object in the IUT that contains a writable property designated P1. An attempt will be made to write to this property using a datatype that the IUT supports but which is not compliant with the property definition given by the BACnet standard.

Configuration Requirements: The value to be written shall not be of a datatype which is compliant with the property definition. If no object supports writable properties, then this test shall be omitted.

Test Steps:

1. READ X = (Object1), P1
2. TRANSMIT WriteProperty-Request,
'Object Identifier' = Object1,
'Property Identifier' = P1,
'Property Value' = (any value with an invalid datatype)
3. RECEIVE (BACnet-Error PDU,
Error Class =PROPERTY,

© 2026 by BACnet International. All rights reserved.

Error Code = INVALID_DATATYPE)
| (BACnet-Reject-PDU
~~Reject Reason = INVALID_PARAMETER_DATATYPE)~~
~~| (BACnet-Reject-PDU~~
~~Reject Reason = INVALID_TAG)~~
Reject Reason = (any reject reason))
4. VERIFY (Object1), P1 = X

9.22.2.X1 Resizing a Writable Array to a Size That Is Not Supported (WriteProperty)

Reason for Change: No test exists for this functionality.

Purpose: This test verifies the IUT can execute WriteProperty requests which would resize an array to a size that is not supported.

Test Concept: The TD shall attempt to resize a resizable array in the IUT, to a size which is not supported by the IUT. An error is returned and TD verifies that the array is unchanged.

Configuration Requirement: O1 is any object in the IUT database having array property P1 having size X.

Test Steps:

1. VERIFY X = (P1, Array Index = 0)
2. IF (X > 0) THEN
3. READ V = P1
4. TRANSMIT WriteProperty-Request,
'Object Identifier' = O1,
'Property Identifier' = P1
'Property Array Index' = 0
'Property Value' = (a value which would resize the array to a size which is unsupported by the IUT for P1)
5. RECEIVE BACnet-Error-PDU,
'Error Class' = PROPERTY,
'Error Code' = INVALID_ARRAY_SIZE | INVALID_ARRAY_INDEX | VALUE_OUT_OF_RANGE,
6. TRANSMIT WriteProperty-Request,
'Object Identifier' = O1,
'Property Identifier' = P1,
'Property Value' = (an array consisting of the number of elements which would resize the array to a size which is unsupported by the IUT for P1)
7. RECEIVE BACnet-Error-PDU,
'Error Class' = PROPERTY,
'Error Code' = INVALID_ARRAY_SIZE | INVALID_ARRAY_INDEX | VALUE_OUT_OF_RANGE,
8. IF (X>0) THEN
9. VERIFY P1 = V
10. ELSE
11. VERIFY (P1 = 0, Array Index = 0)

9.23 WritePropertyMultiple Service Execution Tests

9.23.1 Positive WritePropertyMultiple Service Execution Tests

9.23.1.4 Writing Multiple Properties to Multiple Objects

Reason for Change: Swap which the objects in the VERIFY steps 8 and 9 so the match with the property being checked, per BTL-CR-0611.

© 2026 by BACnet International. All rights reserved.

Purpose: To verify the ability to write multiple properties to multiple objects.

Test Concept: This test case attempts to write properties, P1 and P2, that reside in Object1, and properties P3 and P4 that reside in Object2. P1, P2, P3 and P4 are not commandable properties. If four such writable properties do not exist the test can be modified to write to an array property or to a commandable property with a write priority high enough to ensure that the commandable property's value will change.

Test Steps:

1. READ X = (Object1), P1
2. READ Y = (Object1), P2
3. READ Z = (Object2), P3
4. READ A = (Object2), P4
5. TRANSMIT WritePropertyMultiple-Request,
   'Object Identifier' = Object1,
   'Property Identifier' = P1,
   'Property Value' = (V1, any valid value of the appropriate datatype for this property subject to the restrictions specified in the EPICS as defined in 4.4.2, except the value X read in step 1)
   'Property Identifier' = P2,
   'Property Value' = (V2, any valid value of the appropriate datatype for this property subject to the restrictions specified in the EPICS as defined in 4.4.2, except the value Y read in step 2)
   'Object Identifier' = Object2,
   'Property Identifier' = P3,
   'Property Value' = (V3, any valid value of the appropriate datatype for this property subject to the restrictions specified in the EPICS as defined in 4.4.2, except the value Z read in step 3)
   'Object Identifier' = Object2,
   'Property Identifier' = P4,
   'Property Value' = (V4, any valid value of the appropriate datatype for this property subject to the restrictions specified in the EPICS as defined in 4.4.2, except the value A read in step 4)
6. RECEIVE BACnet-SimpleACK-PDU
7. VERIFY (Object1), P1 = V1
8. VERIFY (~~Object2~~*Object1*), P2 = V2
9. VERIFY (~~Object1~~*Object2*), P3 = V3
10. VERIFY (Object2), P4 = V4

9.23.2 Negative WritePropertyMultiple Service Execution Tests

9.23.2.4 Writing Non-Array Properties with an Array Index

Reason for Change: 135-2020ci-6 clarifies that 'Property Array Index' is required when present in the request even if the property is not an array.

Purpose: This test case verifies that the IUT can execute WritePropertyMultiple service requests when the property value is not an array but an array index is included in the service request. ~~This test shall only be performed if Protocol_Revision is present and has a value greater than or equal to 4.~~

Test Concept: The TD shall select an object, designated Object1, in the IUT that contains a writable scalar property designated P1 having a value X. An attempt will be made to write to this property using an array index, Y. If no suitable object can be found, then this test shall be omitted.

Configuration Requirements: If the IUT supports any writable properties that are scalars, it shall be configured with at least one such property that can be used for this test.

© 2026 by BACnet International. All rights reserved.

Test Steps:
1. READ X = (Object1), P1
2. TRANSMIT WritePropertyMultiple-Request,
   'Object Identifier' = Object1,
   'Property Identifier' = P1,
   'Property Value' = (any valid value of the appropriate datatype for this property subject to the restrictions specified in the EPICS as defined in 4.4.2, except the value X read in step 1)
   `Property Array Index' = (Y, any positive integer)
3. RECEIVE WritePropertyMultiple-Error,
   Error Class = PROPERTY,
   Error Code = PROPERTY_IS_NOT_AN_ARRAY,
   ~~objectIdentifier~~ *'Object Identifier'* = Object1,
   ~~propertyIdentifier~~ *'Property Identifier'* = P1*,*
   'Property Array Index' = Y
4. VERIFY (Object1), P1 = X

9.23.2.5 Writing Array Properties with an Array Index that is *Greater Than The Length Of The Array*~~Out of Range~~

Reason for Change: Update to account for new language in 135-2020ci-6.

Purpose: This test case verifies that the IUT can execute WritePropertyMultiple service requests when the requested property value is an array but the array index is *greater than the length of the array.*~~out of range.~~ This test shall only be performed if Protocol_Revision is present and has a value greater than or equal to 4.
Test Concept: The TD shall select an object, designated Object1, in the IUT that contains a writable array property designated P1 having a value X. An attempt will be made to write to this property using an array index Y, that is *greater than the length of the array.*~~out of range.~~ If no suitable object can be found, then this test shall be omitted.

Configuration Requirements: If the IUT supports any writable properties that are arrays, it shall be configured with at least one such property that can be used for this test.

Test Steps:

1. READ X = (Object1), P1
2. TRANSMIT WritePropertyMultiple-Request,
   'Object Identifier' = Object1,
   'Property Identifier' = P1,
   'Property Value' = (any valid value of the appropriate datatype for this property subject to the restrictions specified in the EPICS as defined in 4.4.2, except the value X ~~read in step 1~~)
   ~~`~~'Property Array Index' = (Y, any value that is *greater than the length of the array)* ~~larger than the supported size of the array)~~
3. RECEIVE WritePropertyMultiple-Error,
   'Error Class' = PROPERTY,
   'Error Code' = INVALID_ARRAY_INDEX,
   'Object Identifier' = Object1,
   'Property Identifier' = P1,
   'Property Array Index' = Y
4. VERIFY (Object1), P1 = X

9.23.2.6 Writing with a Property Value Having the Wrong Datatype

Reason for Change: Cleanup reject codes.

Purpose: This test case verifies that the IUT correctly responds to an attempt to write a property value that has an invalid datatype.

© 2026 by BACnet International. All rights reserved.

Test Concept: The TD shall select an object, designated Object1, in the IUT that contains a writable property designated P1. An attempt will be made to write to this property using a datatype that the IUT supports but which is not compliant with the property definition given by the BACnet standard

Configuration Requirements: The value to be written shall not be of a datatype which is compliant with the property definition. If no object supports writable properties, then this test shall be omitted.

Test Steps:

1. READ X = (Object1), P1
2. TRANSMIT WritePropertyMultiple-Request,
   'Object Identifier' = Object1,
   'Property Identifier' = P1,
   'Property Value' = (any value with an invalid datatype)
3. RECEIVE
   WritePropertyMultiple-Error,
   Error Class = PROPERTY,
   Error Code = INVALID_DATATYPE,
   objectIdentifier = Object1,
   propertyIdentifier = P1
   ~~| BACnet-Reject-PDU~~
   ~~'Reject Reason' = INVALID_PARAMETER_DATATYPE~~
   ~~| BACnet-Reject-PDU~~
   ~~'Reject Reason' = INVALID_TAG~~
   | *(BACnet-Reject-PDU*
   Reject Reason = (any reject reason))
4. VERIFY (Object1), P1 = X

9.23.2.12 WritePropertyMultiple Reject Test

Reason for Change: Fix Test Concept to define O1 and O2.

Purpose: This test case verifies that the IUT does not send a Reject-PDU after applying part of a WritePropertyMultiple.

Test Concept: *Object, O1, containing writable property, P1 and object O2, containing writable property, P2*~~Two writable properties, P1 and P2~~ are written~~ to the IUT~~ but the portion of the WritePropertyMultiple specifying P2 is made invalid by omitting the 'Property Value' parameter. If the IUT returns a Reject, then the value of the first property is checked to ensure it has not changed.

Test Steps:

1. READ OldValue = O1, P1
2. TRANSMIT WritePropertyMultiple-Request,
   'Object Identifier' = O1,
   'Property Identifier' = P1,
   'Property Value' = (NewValue: any value other than OldValue that would be accepted by the IUT for P1)
   'Object Identifier' = O2,
   'Property Identifier' = P2
   -- *'Property Value' = (this field is missing including the opening and closing tags)*
3. RECEIVE
   WritePropertyMultiple-Error,
   'Error Class' = SERVICES,
   'Error Code' = INVALID_TAG
   'Object Identifier' = O2
   'Property Identifier' = P2) |

© 2026 by BACnet International. All rights reserved.

~~4. RECEIVE BACnet Reject-PDU,~~
~~*(BACnet-Reject-PDU*~~
~~'Reject Reason' = INVALID_TAG |~~
~~MISSING_REQUIRED_PARAMETER |~~
~~INCONSISTENT_PARAMETERS |~~
~~INVALID_PARAMETER_DATA_TYPE |~~
~~TOO_MANY_ARGUMENTS)~~
~~*Reject Reason = (any reject reason))*~~
~~*5*~~*4*. IF (a WritePropertyMultiple-Error was received in step 3) THEN
~~*6*~~*5*. VERIFY (O1), P1 = NewValue
ELSE -- a Reject-PDU was received
~~*7*~~*6*. VERIFY (O1), P1 = OldValue

9.23.2.13 Resizing a Writable Fixed Size Array Property Using WritePropertyMultiple Service

Reason for Change: Making this test be consistent with 9.22.2.10 by adding WRITE_ACCESS_DENIED as allowed error. *Updated to allow INVALID_ARRAY_SIZE from ci-6.*

Purpose: This test case verifies that the IUT correctly responds to an attempt to resize a writable fixed size array property using WritePropertyMultiple service.

Test Concept: Select an object(O1) in the IUT that contains a writable array property of a fixed size. This property is designated P1. If no suitable object can be found, then this test shall be omitted.

Test Steps:

1. READ X = (O1), P1, ARRAY INDEX = 0
2. TRANSMIT WritePropertyMultiple-Request,
   'Object Identifier' = O1,
   'Property Identifier' = P1,
   'Property Value' = (Entire Array with any valid value greater than Array Size X)
3. RECEIVE WritePropertyMultiple-Error,
   'Error Class' = PROPERTY,
   'Error Code' = INVALID_ARRAY_INDEX | VALUE_OUT_OF_RANGE | *INVALID_ARRAY_SIZE*,
   'ObjectIdentifier' = O1,
   'PropertyIdentifier' = P1
4. VERIFY (O1), P1= X, ARRAY INDEX = 0
5. TRANSMIT WritePropertyMultiple-Request,
   'Object Identifier' = O1,
   'Property Identifier' = P1,
   'Property Value' = (Entire Array with any valid value less than Array Size X)
6. RECEIVE WritePropertyMultiple-Error,
   'Error Class' = PROPERTY,
   'Error Code' = INVALID_ARRAY_INDEX | VALUE_OUT_OF_RANGE | *INVALID_ARRAY_SIZE,*
   'ObjectIdentifier' = O1,
   'PropertyIdentifier' = P1
7. VERIFY (O1), P1= X, ARRAY INDEX = 0
8. TRANSMIT WritePropertyMultiple-Request,
   'Object Identifier' = O1,
   'Property Identifier' = P1,
   'Property Value' = (any valid value greater than Array Size X),
   'Property Array Index' = 0
9. RECEIVE WritePropertyMultiple-Error,
   'Error Class' = PROPERTY,
   'Error Code' = INVALID_ARRAY_INDEX | VALUE_OUT_OF_RANGE | *WRITE_ACCESS_DENIED* | *INVALID_ARRAY_SIZE*,

© 2026 by BACnet International. All rights reserved.

'ObjectIdentifier' = O1,
'PropertyIdentifier' = P1
'Property Array Index'=0
10. VERIFY (O1), P1= X, ARRAY INDEX = 0
11. TRANSMIT WritePropertyMultiple-Request,
'Object Identifier' = O1,
'Property Identifier' = P1,
'Property Value' = (any valid value less than Array Size X),
'Property Array Index' = 0
12. RECEIVE WritePropertyMultiple-Error,
'Error Class' = PROPERTY,
'Error Code' = INVALID_ARRAY_INDEX | VALUE_OUT_OF_RANGE | *WRITE_ACCESS_DENIED* | *INVALID_ARRAY_SIZE*,
'ObjectIdentifier' = O1,
'PropertyIdentifier' = P1
'Property Array Index'= 0
13. VERIFY (O1), P1= X, ARRAY INDEX = 0

9.23.2.14 Writing First Element of 'List of Write Access Specifications' with Object Access Error

Reason For Change: Updated the test to remove support of the UNSUPPORTED_OBJECT_TYPE error code for devices supporting PR 27 or higher.

Purpose: To verify the ability to correctly execute a WritePropertyMultiple service request for which the first element of the 'List of Write Access Specifications' contains a specification for an unsupported object and all writes after the first failed write attempt do not take place.

Test Concept: An attempt is made to write to a single property in two different objects. The first object is not supported. The second object is supported, and the property is writable. The objective is to verify that an appropriate error response is returned and that all writes after the first failed write attempt do not take place.

Configuration Requirements: If the IUT supports any writable scalar properties that are not commandable it shall be configured with one for use in this test. If no such properties are supported the IUT shall be configured with a writable array or commandable property and the test steps modified to account for this variation. In the test description O2 and P2 will be used to designate the writable object and property having value X used for this test. The designation ~~Bad Object~~ *BadObject* will be used to indicate an object that is not supported or not present in IUT databaseP1 is any valid Property Identifier.

Test Steps:
1. ~~VERIFY (O2), P2 = X~~ *READ X = (O2), P2*
2. TRANSMIT WritePropertyMultiple-Request,
'Object Identifier' = BadObject,
'Property Identifier' = P1,
'Property Value' = (any valid value of the appropriate datatype for *P1*~~this property subject to the restrictions specified in the EPICS as defined in 4.4.2~~)
'Object Identifier' = O2,
'Property Identifier' = P2,
'Property Value' = (any valid value not equal to X),
3. IF (Protocol_Revision < 27) THEN
~~3~~*4*. RECEIVE WritePropertyMultiple-Error,
'Error Class' = OBJECT,
'Error Code' = *(*UNKNOWN_OBJECT | *UNSUPPORTED_OBJECT_TYPE)*,
'Object Identifier' = BadObject,
'Property Identifier' = P1
~~'Property Identifier' = P1 |~~
~~(RECEIVE WritePropertyMultiple-Error,~~

© 2026 by BACnet International. All rights reserved.

~~'Error Class' = OBJECT,~~
~~'Error Code' = UNSUPPORTED_OBJECT_TYPE,~~
~~'Object Identifier' = BadObject,~~
~~'Property Identifier' = P1)~~
ELSE
5. *RECEIVE WritePropertyMultiple-Error,*
'Error Class' = OBJECT,
'Error Code' = UNKNOWN_OBJECT,
'Object Identifier' = BadObject,
'Property Identifier' = P1
~~4~~*6.* VERIFY (O2), P2 = X

9.23.2.15 Writing First Element of 'List of Write Access Specifications' with a Write Access Error

Reason For Change: Replace VERIFY with READ.

Purpose: To verify the ability to correctly execute a WritePropertyMultiple service request for which the first element of the 'List of Write Access Specifications' contains a specification for a read only property and all writes after the first failed write attempt do not take place.

Test Concept: An attempt is made to write to two properties in a single object. The first property is supported but read only. The second property is supported and writable. The objective is to verify that an appropriate error response is returned and that all writes after the first failed write attempt do not take place.

Configuration Requirements: If the IUT supports any writable scalar properties that are not commandable, it shall be configured with one for use in this test. If no such properties are supported, the IUT shall be configured with a writable array or commandable property and the test steps modified to account for this variation. In the test description, O1 will be used to designate the object, P1 the read only property having value X, P2 the writable property having value Y used for this test.

Test Steps:
1. ~~VERIFY (O1), P1= X~~ *READ X = (O1), P1*
2. ~~VERIFY (O1), P2=Y~~ *READ Y = (O1), P2*
3. TRANSMIT WritePropertyMultiple-Request,
'Object Identifier' = O1,
'Property Identifier' = ~~P1~~ *(P1, a read-only property in O1)*,
'Property Value' = X,
'Property Identifier' = P2,
'Property Value' = (any valid value not equal to Y)
4. RECEIVE WritePropertyMultiple-Error,
'Error Class' = PROPERTY,
'Error Code' = WRITE_ACCESS_DENIED,
'Object Identifier' = O1,
'Property Identifier' = P1
5. VERIFY (O1), P1 = X
~~5~~*6.* VERIFY (O1), P2 = Y

9.23.2.16 WritePropertyMultiple Reject Test for First Element of 'List of Write Access Specifications'

Reason for Change: Add Read steps 1&2 and comment in step 3. *Cleanup reject codes.*

Purpose: This test case verifies that if IUT does sends a Reject-PDU or Error-PDU then the write attempt for the remaining element of 'List of Write Access Specifications' do not take place.

© 2026 by BACnet International. All rights reserved.

Test Concept: Object, O1, contains a writable property, P1 and object O2, contains a writable property, P2. P1 having value X and P2 having value Y are written to the IUT but the portion of the WritePropertyMultiple specifying P1 is made invalid by omitting the 'Property Value' parameter. The value of the properties are checked to ensure they have not changed.

Test Steps:

1. READ X = (O1), P1
2. READ Y = (O2), P2
~~1~~*3.* TRANSMIT WritePropertyMultiple-Request,
'Object Identifier' = O1,
'Property Identifier' = P1,
-- 'Property Value' = (this field is missing including the opening and closing tags)
'Object Identifier' = O2,
'Property Identifier' = P2
'Property Value' = (Any valid value not equal to Y))
~~2~~*4.* RECEIVE (WritePropertyMultiple-Error,
'Error Class' = SERVICES,
'Error Code' = INVALID_TAG
'Object Identifier' = O1
'Property Identifier' = P1) |
~~(BACnet-Reject-PDU,~~
~~'Reject Reason' = INVALID_TAG |~~
~~MISSING_REQUIRED_PARAMETER |~~
~~INCONSISTENT_PARAMETERS |~~
~~INVALID_PARAMETER_DATA_TYPE |~~
(BACnet-Reject-PDU
Reject Reason = (any reject reason))
~~3~~*5.* VERIFY (O1), P1 = X
~~4~~*6.* VERIFY (O2), P2 = Y

9.23.2.17 Writing First Element of 'List of Write Access Specifications' with a Property Access Error

Reason For Change: Replace VERIFY with READ.

Purpose: To verify the ability to correctly execute a WritePropertyMultiple service request for which the first element of the 'List of Write Access Specifications' contains a specification for an unsupported property and all writes after the first failed write attempt do not take place.

Test Concept: An attempt is made to write to two properties in a single object. The first property is not supported for this object. The second property is supported for this object and writable. The objective is to verify that an appropriate error response is returned and that all writes after the first failed write attempt do not take place.

Configuration Requirements: If the IUT supports any writable scalar properties that are not commandable, it shall be configured with one for use in this test. If no such properties are supported, the IUT shall be configured with a writable array or commandable property and the test steps modified to account for this variation. In the test description, O1 will be used to designate the object, P1 the unsupported property, and P2 the writable property having value X used.

Test Steps:
1. ~~VERIFY (O1), P2 = X~~ *READ X = (O1), P2*
2. TRANSMIT WritePropertyMultiple-Request,
'Object Identifier' = O1,
'Property Identifier' = ~~P1~~*(P1, an unsupported property for object O1)*,
'Property Value' = (any valid value of the appropriate datatype for *P1*~~this property subject to the restrictions specified in the EPICS as defined in 4.4.2~~)
'Property Identifier' = P2,

© 2026 by BACnet International. All rights reserved.

'Property Value' = (any valid value not equal to X),
3. RECEIVE WritePropertyMultiple-Error,
'Error Class' = PROPERTY,
'Error Code' = UNKNOWN_PROPERTY,
'Object Identifier' = O1,
'Property Identifier' = P1
4. VERIFY (O1), P2 = X

9.23.2.19 Date Non-Pattern Properties Test using WritePropertyMultiple Service

Reason for Change: Update Test Concept to include meaning of O1. *Cleanup reject codes.*

Purpose: To verify that the property being tested does not accept special date field values.

Test Concept: *O1 is the object being tested.* The property being tested, P1, is written with each of the special date field values to ensure that the property does not accept them. A date is selected which is within the date range that the IUT will accept for the property. The value, V1, written to the property is the date D1 with one of its fields replaced with one of the date special values. If the property is a complex datatype, the other fields in the value shall be set within the range accepted by the IUT. ~~This test shall only be applied to devices claiming Protocol_Revision 11 or higher.~~

Notes to Tester: if P1 is an array, then a non-zero array index may be provided in the TRANSMIT and the same array index observed in the WritePropertyMultiple-Error.

Test Steps:
1. REPEAT SV = (year unspecified, month unspecified, day of month unspecified,
day of week unspecified, odd months, even months, last day of month,
even days, odd days) DO {
2. TRANSMIT WritePropertyMultiple-Request
'Object Identifier' = O1,
'Property Identifier' = P1,
'Property Value' = (V1 updated with the special value SV)
3. RECEIVE
WritePropertyMultiple-Error,
'Error Class' = PROPERTY,
'Error Code' = VALUE_OUT_OF_RANGE,
'Object Identifier' = O1,
'Property Identifier' = P1)
| (BACnet-Reject-PDU
~~'Reject Reason' = INVALID_PARAMETER_DATATYPE)~~
~~| (BACnet-Reject-PDU~~
~~'Reject Reason'= INVALID_TAG)~~
Reject Reason = (any reject reason))
}

9.23.2.20 Time Non-Pattern Properties Test using WritePropertyMultiple Service

Reason for Change: Update Test Concept to include meaning of O1. *Cleanup reject codes.*

Purpose: To verify that the property being tested does not accept special time field values.

Test Concept: *O1 is the object being tested.* The property being tested, P1, is written with each of the special time field values to ensure that the property does not accept them. A time is selected which is within the time range that the IUT will accept for the property. The value, V1, written to the property is the time T1 with one of its fields replaced with one of the time special values. If the property is a complex datatype, the other fields in the value shall be set within the range accepted by the IUT~~. This test shall only be applied to devices claiming Protocol_Revision 11 or higher.~~

© 2026 by BACnet International. All rights reserved.

Notes to Tester: if P1 is an array, then a non-zero array index may be provided in the TRANSMIT and the same array index observed in the WritePropertyMultiple-Error.

Test Steps:

1. REPEAT SV = (hour unspecified, minute unspecified, second unspecified, hundredths unspecified) DO {
2. TRANSMIT WritePropertyMultiple-Request
'Object Identifier' = O1,
'Property Identifier' = P1,
'Property Value' = (V1 updated with the special value SV)
3. RECEIVE
(WritePropertyMultiple-Error,
'Error Class' = PROPERTY,
'Error Code' = VALUE_OUT_OF_RANGE,
'Object Identifier' = Object1,
'Property Identifier' = P1)
| (BACnet-Reject-PDU
~~'Reject Reason' = INVALID_PARAMETER_DATATYPE)~~
~~| (BACnet-Reject-PDU~~
~~'Reject Reason'= INVALID_TAG)~~
Reject Reason = (any reject reason))
}

9.23.2.21 DateTime Non-Pattern Properties Test using WritePropertyMultiple Service

Reason for Change: Update Test Concept to include meaning of O1. Remove day of week test based on IR. *Cleanup reject codes.*

Purpose: To verify that the property being tested does not accept special date field values.

Test Concept: *O1 is the object being tested*. The property being tested, P_1, is written with each of the special datetime field values to ensure that the property does not accept them. A datetime DT_1 is selected which is within the range that the IUT will accept for the property. The value, V_1, written to the property is the datetime DT_1 with one of its fields replaced with one of the date or time special values. If the property is a complex datatype, the other fields in the value shall be set within the range accepted by the IUT. *It is a local matter whether the device accepts or rejects an invalid day of week field so it is not tested.*~~This test shall only be applied to devices claiming Protocol_Revision 11 or higher.~~

~~Notes to Tester: if P1 is an array, then a non-zero array index may be provided in the TRANSMIT and the same array index observed in the WritePropertyMultiple-Error.~~

Test Steps:
1. REPEAT SV = (year unspecified, month unspecified, day of month unspecified, ~~day of week unspecified,~~
odd months, even months, last day of month, even days, odd days, hour unspecified, minute unspecified,
second unspecified, hundredths unspecified) DO {
2. TRANSMIT WritePropertyMultiple-Request,
'Object Identifier' = O1,
'Property Identifier' = P1,
'Property Value' = (DT_1 updated with the special value SV)
~~3. RECEIVE WritePropertyMultiple-Error,~~
~~'Error Class' = PROPERTY,~~
~~'Error Code' = VALUE_OUT_OF_RANGE,~~
~~'Object Identifier' = Object1,~~
~~'Property Identifier' = P1)~~
~~| (BACnet-Reject-PDU~~
~~'Reject Reason' = INVALID_PARAMETER_DATATYPE)~~
~~| (BACnet-Reject-PDU~~

© 2026 by BACnet International. All rights reserved.

~~'Reject Reason'= INVALID_TAG)~~

3. RECEIVE
(WritePropertyMultiple-Error,
'Error Class' = PROPERTY,
'Error Code' = VALUE_OUT_OF_RANGE,
'Object Identifier' = Object1,
'Property Identifier' = P1)
| (BACnet-Reject-PDU
Reject Reason = (any reject reason))
}

9.23.2.22 BACnetDateRange Non-Pattern Properties Test using WritePropertyMultiple Service

Reason for Change: *Remove day of week test based on IR and reject cleanup*.

Purpose: To verify that the property being tested does not accept special date field values, except for fully unspecified start of the range or fully unspecified end of the range, or both.

Test Concept: O1 is the object being tested. A BACnetDateRange property, or property that is a complex datatype containing BACnetDateRange P1 is written with each of the special field values to ensure that the property does not accept them. Each half of the dateRange DR1 is selected so it is within the range that the IUT will accept for the property. The value, V1 written to the property is the dateRange DR1 with one of its fields replaced with one of the date special values. If the property is a complex datatype, the other fields in the value shall be set within the range accepted by the IUT. *It is a local matter whether the device accepts or rejects an invalid day of week field so it is not tested.* ~~This test shall only be applied to devices claiming Protocol_Revision 11 or higher.~~

Notes to Tester: if P1 is an array, then a non-zero array index may be provided in the TRANSMIT and the same array index observed in the WritePropertyMultiple-Error.

Test Steps:

1. REPEAT SV = (year unspecified, month unspecified, day of month unspecified,
~~day of week unspecified,~~ odd months, even months, last day of month,
even days, odd days) DO {
2. TRANSMIT WritePropertyMultiple-Request,
'Object Identifier' = O1,
'Property Identifier' = P1,
'Property Value' = (DR1 with startDate updated with special value SV)
3. RECEIVE
(WritePropertyMultiple-Error,
'Error Class' = PROPERTY,
'Error Code' = VALUE_OUT_OF_RANGE,
'Object Identifier' = Object1,
'Property Identifier' = P1)
| (BACnet-Reject-PDU
~~'Reject Reason' = INVALID_PARAMETER_DATATYPE)~~
~~| (BACnet-Reject-PDU~~
~~'Reject Reason'= INVALID_TAG)~~
Reject Reason = (any reject reason))
4. TRANSMIT WritePropertyMultiple-Request,
'Object Identifier' = O1,
'Property Identifier' = P1,
'Property Value' = (DR1 with endDate updated with special value SV)
5. RECEIVE
(WritePropertyMultiple-Error,
'Error Class' = PROPERTY,
'Error Code' = VALUE_OUT_OF_RANGE,

© 2026 by BACnet International. All rights reserved.

'Object Identifier' = Object1,
'Property Identifier' = P1)
| (BACnet-Reject-PDU
~~'Reject Reason' = INVALID_PARAMETER_DATATYPE)~~
~~| (BACnet-Reject-PDU~~
~~'Reject Reason'= INVALID_TAG)~~
Reject Reason = (any reject reason))
}

9.23.2.X1 Resizing a Writable Array to a Size That Is Not Supported (WritePropertyMultiple)

Reason for Change: No test exists for this functionality.

Purpose: This test verifies the IUT can execute WritePropertyMultiple requests which would resize an array to a size that is not supported.

Test Concept: The TD shall attempt to resize a resizable array in the IUT, to a size which is not supported by the IUT. An error is returned and TD verifies that the array is unchanged.

Configuration Requirement: O1 is any object in the IUT database having array property P1 having size X.

Test Steps:

1. VERIFY X = (P1, Array Index = 0)
2. IF (X > 0) THEN
3. READ V = P1
4. TRANSMIT WritePropertyMultiple-Request,
'Object Identifier' = O1,
'Property Identifier' = P1
'Property Array Index' = 0
'Property Value' = (a value which would resize the array to a size which is unsupported by the IUT for P1)
5. RECEIVE WritePropertyMultiple-Error,
'Error Class' = PROPERTY,
'Error Code' = INVALID_ARRAY_SIZE | INVALID_ARRAY_INDEX | VALUE_OUT_OF_RANGE,
'Object Identifier' = O1,
'Property Identifier' = P1,
'Property Array Index' = 0
6. TRANSMIT WritePropertyMultiple-Request,
'Object Identifier' = O1,
'Property Identifier' = P1,
'Property Value' = (an array consisting of the number of elements which would resize the array to a size which is unsupported by the IUT for P1)
7. RECEIVE WritePropertyMultiple-Error,
'Error Class' = PROPERTY,
'Error Code' = INVALID_ARRAY_SIZE | INVALID_ARRAY_INDEX | VALUE_OUT_OF_RANGE,
'Object Identifier' = O1,
'Property Identifier' = P1
8. IF (X > 0) THEN
9. VERIFY V = P1
10. ELSE
11. VERIFY 0 = (P1, Array Index = 0)

© 2026 by BACnet International. All rights reserved.

9.24 DeviceCommunicationControl Service Execution Test

9.24.1 Positive DeviceCommunicationControl Service Execution Tests

[Remove test 9.24.1.1]

[Remove test 9.24.1.2]

[Remove test 9.24.1.3]

[Remove test 9.24.1.4]

[Remove test 9.24.1.5]

[Remove test 9.24.1.6]

[Remove test 9.24.1.7]

[Remove test 9.24.1.8]

[Remove test 9.24.1.9]

[Remove test 9.24.1.10]

[Remove test 9.24.1.11]

[Remove test 9.24.1.12]

9.24.1.X1 DeviceCommunicationControl Enable-Disable Test

Reason For Change: If the IUT does not support initiation, many of the tests are skipped so other aspects of DCC are not tested. This test validates simple and more complex IUTs.

Purpose: To verify the IUT correctly executes the actions of DISABLE, DISABLE_INITIATION, and ENABLE.

Test Concept: Send DISABLE and ENABLE and verify the IUT accepts the commands. Send DISABLE_INITIATION and verify read requests, Who-Is/I-Am, and Who-Has/I-Have function and the IUT does not initiate any other requests.

Notes to Tester: Execute this test with the Time Duration parameter present. If the IUT does not require or can be configured to not require a Password, execute this test with any string in the Password parameter. If the IUT requires a Password, execute this test with a valid password in the Password parameter.

Test Steps:

-- test DISABLE
1. IF (Protocol_Revision < 20) THEN {
2. TRANSMIT DeviceCommunicationControl-Request,
'Time Duration' = (any valid value long enough to execute this test),
'Enable/Disable' = DISABLE,
'Password' = (an appropriate password)
3. RECEIVE BACnet-SimpleACK-PDU
4. TRANSMIT DeviceCommunicationControl-Request,
'Enable/Disable' = ENABLE,
'Password' = (an appropriate password)
5. RECEIVE BACnet-SimpleACK-PDU
6. CHECK (Verify that the IUT has not transmitted any APDUs since the acknowledgment in step 2)
}
ELSE {
7. TRANSMIT DeviceCommunicationControl-Request,
'Time Duration' = (any valid value long enough to execute this test),
'Enable/Disable' = DISABLE,
'Password' = (an appropriate password)
8. RECEIVE BACnet-Error-PDU

© 2026 by BACnet International. All rights reserved.

```
                Error Class = SERVICES,
                Error Code = SERVICE_REQUEST_DENIED
        }

 -- test DISABLE_INITIATION
9.    TRANSMIT DeviceCommunicationControl-Request,
            'Time Duration' =       (any valid value long enough to execute this test),
            'Enable/Disable' =      DISABLE_INITIATION,
            'Password' =        (an appropriate password)
10.   RECEIVE BACnet-SimpleACK-PDU
 -- test the IUT still responds to requests
11.   VERIFY (any supported property) = (any valid value)
-- test I-Am
12.   TRANSMIT Who-Is-Request
13.   RECEIVE I-Am-Request
14.   IF (Protocol_Revision >= 30) THEN {
            -- test I-Have
15.         TRANSMIT Who-Has-Request
                'Object Name' = (O1, any object in the IUT)
16.         RECEIVE I-Have-Request
                'Device Identifier' = (the IUT's Device object),
                'Object Identifier' = O1,
                'Object Name' = (O1's object name)
        }
17.   IF (the IUT can initiate a message other than I-Am in response to Who-Is,
                I-Have in response to Who-Has, or
                ConfirmedAuditNotification and UnconfirmedAuditNotification requests) THEN {
            -- test the IUT does not initiate any requests other than specified
18.         MAKE (do something that would normally cause the IUT to initiate a message)
19.         WAIT (an arbitrary time > Internal Processing Fail Time selected by the tester)
20.         CHECK (Verify that the IUT has not transmitted any APDUs)
        }
21.   TRANSMIT DeviceCommunicationControl-Request,
            'Enable/Disable' =      ENABLE,
            'Password' =        (an appropriate password)
22.   RECEIVE BACnet-SimpleACK-PDU
```

9.24.1.X2 DeviceCommunicationControl Restored with Time Expired

Reason For Change: Replaces 9.24.1.3, 9.24.1.8, 9.24.1.9.

Purpose: To verify the correct execution of the DeviceCommunicationControl request service procedure when finite time duration is specified.

Test Steps:

```
1.    TRANSMIT DeviceCommunicationControl-Request,
            'Time Duration' =       (a value T > 1, in minutes, selected by the tester),
            'Enable/Disable' =      (parameter configured for this test),
            'Password' =        (absent or an appropriate password)
2.    RECEIVE BACnet-SimpleACK-PDU
3.    WAIT (T)
4.    VERIFY (any supported property) = (any valid value)
```

© 2026 by BACnet International. All rights reserved.

9.24.1.X3 DeviceCommunicationControl Restored by DeviceCommunicationControl

Reason For Change: Replaces 9.24.1.1, 9.24.1.4, 9.24.1.6, 9.24.1.10.

Purpose: To verify the correct execution of the DeviceCommunicationControl request service procedure communication is restored using the DeviceCommunicationControl service.

Test Steps:

1. TRANSMIT DeviceCommunicationControl-Request,
   'Time Duration' = (absent or a value T > 1, in minutes, selected by the tester),
   'Enable/Disable' = (parameter configured for this test),
   'Password' = (absent or an appropriate password)
2. RECEIVE BACnet-SimpleACK-PDU
3. TRANSMIT DeviceCommunicationControl-Request,
   'Enable/Disable' = ENABLE,
   'Password' = (absent or an appropriate password)
4. RECEIVE BACnet-SimpleACK-PDU
5. VERIFY (any supported property) = (any valid value)

9.24.1.X4 DeviceCommunicationControl Restored by ReinitializeDevice

Reason For Change: Replaces 9.24.1.2, 9.24.1.5, 9.24.1.7, 9.24.1.12.

Purpose: To verify the correct execution of the DeviceCommunicationControl request service procedure communication is restored using the ReinitializeDevice service.

Test Steps:

1. TRANSMIT DeviceCommunicationControl-Request,
   'Time Duration' = (absent or a value T > 1, in minutes, selected by the tester),
   'Enable/Disable' = (parameter configured for this test),
   'Password' = (absent or an appropriate password)
2. RECEIVE BACnet-SimpleACK-PDU
3. TRANSMIT ReinitializeDevice-Request,
   'Reinitialize State of Device' = WARMSTART,
   'Password' = (absent or an appropriate password)
4. RECEIVE BACnet-SimpleACK-PDU
5. CHECK (Did the IUT perform a WARMSTART?)
6. VERIFY (any supported property) = (any valid value)

9.24.1.X5 DeviceCommunicationControl Password Not Required

Reason For Change: No test for this functionality. Clause 16.1.1.1.3 requires the Password to be ignored if not required.

Purpose: To verify the correct execution of the DeviceCommunicationControl request service procedure when a password is not required.

Test Steps:

1. TRANSMIT DeviceCommunicationControl-Request,
   'Time Duration' = (T, any valid value),
   'Enable/Disable' = DISABLE_INITIATION,
2. RECEIVE BACnet-SimpleACK-PDU
3. WAIT (T)
4. TRANSMIT DeviceCommunicationControl-Request,

© 2026 by BACnet International. All rights reserved.

'Time Duration' = (T, any valid value),
'Enable/Disable' = DISABLE_INITIATION,
'Password' = (any valid arbitrary string)

5. RECEIVE BACnet-SimpleACK-PDU
6. WAIT (T)

9.24.2 Negative DeviceCommunicationControl Service Execution Tests

[Remove test 9.24.2.3]

9.33 Who-Is Service Execution Tests

9.33.2 Execution of Who-Is Service Requests Originating from a Remote Network

The purpose of this test group is to verify the correct execution of the Who-Is request service procedure for messages originating from a remote network. A comprehensive set of variations in Who-Is request parameters is not included in this test group because they are tested in 9.33.1. The tests in this group only represent variations in network layer addressing information.

The IUT is allowed to initiate a Who-Is-Router-To-Network to locate the router to the remote network before initiating the I-Am /I-Have message.

9.33.2.1 General Inquiry, Global Broadcast from a Remote Network

Reason for Change: Clarify the DINFO designations of the expected response.

Purpose: To verify the ability of the IUT to recognize the origin of a globally broadcast Who-Is service request and respond such that the device originating the request receives the response.

Test Steps:

1. TRANSMIT
   DESTINATION = GLOBAL BROADCAST,
   SNET = ~~(any remote network number~~ *X, any remote network number*~~)~~,
   SADR = ~~(any MAC address valid for the specified network~~ *Y, any MAC address valid for the specified network*~~)~~,
   Who-Is-Request
2. BEFORE **Unconfirmed Response Fail Time**
3. RECEIVE
   ~~DESTINATION = GLOBAL BROADCAST | REMOTE BROADCAST (to the network specified by SNET in step 1) | TD ,~~
   DESTINATION = GLOBAL BROADCAST | REMOTE BROADCAST (to network X)| TD (DNET = X, DADR = Y),
   I-Am-Request,
   'I Am Device Identifier' = (the IUT's Device object),
   'Max APDU Length Accepted' = (the value specified in the EPICS),
   'Segmentation Supported' = (the value specified in the EPICS),
   'Vendor Identifier' = (the identifier registered for this vendor)

© 2026 by BACnet International. All rights reserved.

9.33.2.2 General Inquiry, Remote Broadcast

Reason for Change: Clarify the DINFO designations of the expected response.

Purpose: To verify the ability of the IUT to recognize the origin of a remotely broadcast Who-Is service request and respond such that the device originating the request receives the response.

Test Steps:

1. TRANSMIT
   DESTINATION = LOCAL BROADCAST,
   SNET = ~~(any remote network number~~ *X, any remote network number*~~)~~,
   SADR = ~~(any MAC address valid for the specified network~~ *Y, any MAC address valid for the specified network*~~)~~,
   Who-Is-Request
2. BEFORE **Unconfirmed Response Fail Time**
3. RECEIVE
   ~~DESTINATION = GLOBAL BROADCAST | REMOTE BROADCAST (to the network specified by SNET in step 1) | TD ,~~
   DESTINATION = GLOBAL BROADCAST | REMOTE BROADCAST (to network X)| TD (DNET = X, DADR = Y),
   I-Am-Request,
   'I Am Device Identifier' = (the IUT's Device object),
   'Max APDU Length Accepted' = (the value specified in the EPICS),
   'Segmentation Supported' = (the value specified in the EPICS),
   'Vendor Identifier' = (the identifier registered for this vendor)

9.33.2.3 General Inquiry, Directed to a Remote Device

Reason for Change: Clarify the DINFO designations of the expected response.

Purpose: To verify that the IUT responds with an I-Am service that is of the form global broadcast, remote broadcast or unicast.

Test Steps:

1. TRANSMIT
   DESTINATION = IUT,
   SNET = ~~(any remote network number~~ *X, any remote network number*~~)~~,
   SADR = ~~(any MAC address valid for the specified network~~ *Y, any MAC address valid for the specified network*~~)~~,
   Who-Is-Request
2. BEFORE **Unconfirmed Response Fail Time**
3. RECEIVE
   ~~DESTINATION = GLOBAL BROADCAST | REMOTE BROADCAST (to the network specified by SNET in step 1) | TD ,~~
   DESTINATION = GLOBAL BROADCAST | REMOTE BROADCAST (to network X)| TD (DNET = X, DADR = Y),
   I-Am-Request,
   'I Am Device Identifier' = (the IUT's Device object),
   'Max APDU Length Accepted' = (the value specified in the EPICS),
   'Segmentation Supported' = (the value specified in the EPICS),

© 2026 by BACnet International. All rights reserved.

'Vendor Identifier' = (the identifier registered for this vendor)

9.39 General Testing Service Execution

9.39.2 Unsupported Unconfirmed Services Test

Reason for Change: Changed test to verify System_Status vs check for no reset.

Purpose: This test case verifies that the IUT will quietly accept and discard any unconfirmed services that it does not support. When determining the set of services to send to the IUT, the UnconfirmedPrivateTransfer service should be included regardless of whether the IUT supports it or not. The UnconfirmedPrivateTransfer service shall be sent with a vendor ID/Service Number pair not supported by the device.

Configuration Requirements: This test requires that the IUT be placed into a normal operating state ~~in which it will not initiate any requests~~.

Test Steps:

1. READ S1 = System_Status
~~1~~*2.* VERIFY *S1* = ~~System_Status ==~~ OPERATIONAL | OPERATIONAL_READ_ONLY
~~2~~*3.* REPEAT X = (all unconfirmed services that the IUT does not execute) DO {
TRANSMIT X
WHILE (Unconfirmed Response Fail Time) DO {
CHECK (the IUT does not send any packets in response to X)
}
VERIFY System_Status = S1
~~BEFORE **Internal Processing Fail Time***~~
~~CHECK (verify that the IUT did not reset and that the IUT did not send any packets)~~
~~VERIFY System_Status = (the value of System_Status read in step 1)~~
}

~~Passing Result: The IUT does not reset and sends no packets in response to the services.~~

9.42 SubscribeCOVPropertyMultiple Service Execution Tests

9.42.2 Negative SubscribeCOVPropertyMultiple Service Execution Tests

9.42.2.6 Array Index Provided Is *Greater Than The Current Length Of The Array*~~Out Of Range~~

Reason for Change: Update to account for new language in 135-2020ci-6.

Purpose: To verify that the IUT correctly responds to a SubscribeCOVPropertyMultiple request to establish a subscription when the monitored property is an array, but the provided array index is *greater than the current length of the array.*~~outside the range of the array.~~

Test Concept: A subscription for COVM notifications is made which includes a request for an array property P1, with an array index, in object O1 where the IUT supports COVM for P1. All requested subscriptions before P1 in O1 are selected such that they would succeed if P1 in O1 were not in the list. It is verified that the IUT returns the correct error indicating O1 and P1 as the first failed element encountered.

Configuration Requirements: If the IUT does not support COVM on any array properties, then this test shall be skipped.

Test Steps:

1. TRANSMIT SubscribeCOVPropertyMultiple-Request
'Subscriber Process Identifier' = (any valid process identifier),
'Issue Confirmed Notifications' = TRUE | FALSE,

© 2026 by BACnet International. All rights reserved.

'Lifetime' = (any valid lifetime),
'Max Notification Delay' = (any valid value smaller than the lifetime),
'List of COV Subscription Specifications' = (a list of subscriptions for properties in O1 and optionally other objects with P1 being the first property which cannot be subscribed to. The array index included in the entry for P1 shall be larger than the number of entries in P1)

2. RECEIVE BACnet-Error-PDU,
'First-Failed-Subscription' = {
'Monitored Object Identifier' = O1,
'Monitored Property Reference' = P1,
'Error Class' = PROPERTY,
'Error Code' = INVALID_ARRAY_INDEX
}

9.43 Who-Am-I Service Execution Tests

9.43.1 Uses Who-Is to Configure Devices Supporting the Who-Am-I Service

Reason for Change: Fixed fail times.

Purpose: To verify that the IUT can configure a device when it receives a Who-Am-I in response to a Who-Is that it sent.

Test Concept: The IUT sends a Who-Is and TD responds with a Who-Am-I. The IUT then configures the TD's Device object instance number (X) using the You-Are service.

Configuration Requirements: TD has a MAC address but is not configured with a Device object instance number.

Notes to Tester: The IUT may require the tester to specify a Device object instance number to assign TD, using the IUT's software. If the IUT requires this step to occur at the beginning of step 3, then the amount of time it takes the tester to enter this should not be counted towards the fail timer. The destination address used by TD shall be selected such that the IUT will receive the messages.

Test Steps:

1. MAKE (the IUT transmit a Who-Is request to discover devices needing configuration)
2. RECEIVE
DESTINATION = TD | LOCAL BROADCAST | GLOBAL BROADCAST | REMOTE BROADCAST,
Who-Is Request,
'Device Instance Range Low Limit' = 4194303,
'Device Instance Range High Limit' = 4194303
3. TRANSMIT
DESTINATION = IUT | LOCAL BROADCAST | GLOBAL BROADCAST | REMOTE BROADCAST,
Who-Am-I Request,
'Vendor Identifier' = (TD's Vendor_Identifier),
'Model Name' = (any valid Model Name),
'Serial Number' = (any valid Serial Number)
~~4. BEFORE Internal Processing Fail Time~~
4. BEFORE Unconfirmed Response Fail Time
5. RECEIVE
DESTINATION = TD~~| GLOBAL BROADCAST | LOCAL BROADCAST | REMOTE BROADCAST~~,
You-Are Request,
'Vendor Identifier' = (TD's Vendor_Identifier),
'Model Name' = (the Model Name sent by TD in the previous step),
'Serial Number' = (the Serial Number sent by TD in the previous step),
'Device Identifier' = (Device, X),
'Device MAC Address' = (TD's MAC address, or absent)
~~6. IF (TD is not an MS/TP subordinate node)~~
6. BEFORE Unconfirmed Response Fail Time

© 2026 by BACnet International. All rights reserved.

7. TRANSMIT
   DESTINATION = IUT | 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' = (TD's Vendor_Identifier)

9.43.2 Configures Other Device's Object Instance Number

Reason for Change: Fixed fail times.

Purpose: To verify the IUT can configure another Device's object instance number using the You-Are service.

Test Concept: TD sends a Who-Am-I, and the IUT configures it with an appropriate Device object instance number (X).

Configuration Requirements: TD is configured with a MAC address but no Device object instance number. The IUT is not actively discovering devices.

Notes to Tester: The IUT may require the tester to specify a Device object instance number to assign TD, using the IUT's software. If the IUT requires this step to occur between it receiving the Who-Am-I request and sending a You-Are request, then the amount of time it takes the tester to enter this should not be counted towards the fail timer. The destination address used by TD shall be selected such that the IUT will receive the messages.

Test Steps:

1. TRANSMIT
   DESTINATION = IUT | LOCAL BROADCAST | GLOBAL BROADCAST | REMOTE BROADCAST,
   Who-Am-I Request,
   'Vendor Identifier' = (TD's Vendor_Identifier),
   'Model Name' = (any valid Model Name),
   'Serial Number' = (any valid Serial Number)
~~2. BEFORE Internal Processing Fail Time~~
2. BEFORE Unconfirmed Response Fail Time
3. RECEIVE
   DESTINATION = TD~~| LOCAL BROADCAST | GLOBAL BROADCAST | REMOTE BROADCAST~~,
   You-Are Request,
   'Vendor Identifier' = (TD's Vendor_Identifier),
   'Model Name' = (the Model Name sent by TD in the previous step),
   'Serial Number' = (the Serial Number sent by TD in the previous step),
   'Device Identifier' = (Device, X),
   'Device MAC Address' = (TD's MAC address, or absent)
~~4. IF (TD is not an MS/TP subordinate node)~~
4. BEFORE Unconfirmed Response Fail Time
5. TRANSMIT
   DESTINATION = IUT | 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' = (TD's Vendor_Identifier)

© 2026 by BACnet International. All rights reserved.

9.44 You-Are Service Execution Tests

9.44.1 Positive Tests

9.44.1.1 Configurable Device Object Instance Number

Reason for change: Removed MS/TP Subordinate support and removed test cannot be skipped.

Purpose: To verify the IUT will send a Who-Am-I while *the Device object instance number is unconfigured* ~~in the unconfigured state~~.

Test Concept: The IUT is made to send a Who-Am-I while unconfigured. TD configures the IUT's Device object instance number (X).

Configuration Requirements: The IUT is not configured with a Device object instance number. The IUT is configured with a MAC address. ~~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. MAKE (the IUT send a Who-Am-I)
2. RECEIVE
   DESTINATION = TD | LOCAL BROADCAST | GLOBAL BROADCAST | REMOTE BROADCAST,
   Who-Am-I Request,
   'Vendor Identifier' = (the IUT's Vendor_Identifier),
   'Model Name' = (the IUT's Model_Name),
   'Serial Number' = (the IUT's Serial_Number)
3. TRANSMIT
   DESTINATION = IUT ~~| 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' = (Device, X),
   'Device MAC Address' = (IUT's MAC address, or absent)

~~4. IF (the IUT is not an MS/TP subordinate node)~~

~~5~~4. BEFORE Unconfirmed Response Fail Time

~~6~~5. 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)

9.44.1.2 Configurable MAC Address

Reason for change: Removed MS/TP Subordinate support, verify device instance is not changed, and fixed WAIT times. 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.

© 2026 by BACnet International. All rights reserved.

Configuration Requirements: The IUT is not configured with a MAC address *or MAC address (M0)*. 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' = (*M1, any* ~~a~~ valid MAC address *not equal to M0*)

~~2. IF (the IUT is not an MS/TP subordinate node)~~
~~3. BEFORE Internal Processing Fail Time~~

2. BEFORE Unconfirmed Response Fail Time

4*3*. RECEIVE
   SOURCE = M1
   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)

9.44.1.3 Configurable Device Object Instance Number and MAC Address

Reason for change: Removed MS/TP Subordinate support, fixed WAIT times, and verify MAC Address changed.

Purpose: To verify the IUT can be configured with both a Device object instance number and MAC address using the You-Are service.

Test Concept: TD sends a You-Are request to configure the device with both a *new* Device object instance number (X*1*) and *new* a MAC address *(M1)*.

Configuration Requirements: The IUT is ~~not~~ configured with a Device object instance number *(X0) and* ~~or~~ a MAC address *(M0)*. ~~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' = (Device, X*1, any valid value not equal to X0*),
   'Device MAC Address' = (*M1,* any valid MAC address not equal to *M0*)

~~2. IF (the IUT is not an MS/TP subordinate node)~~
~~3. BEFORE Internal Processing Fail Time~~

2. BEFORE Unconfirmed Response Fail Time

4*3*. RECEIVE

© 2026 by BACnet International. All rights reserved.

SOURCE = M1
DESTINATION = TD | LOCAL BROADCAST | GLOBAL BROADCAST | REMOTE BROADCAST,
I-Am Request,
'Device Identifier' = (Device, X*1*),
'Max APDU Length Accepted' = (any valid value),
'Segmentation Supported' = (any valid value),
'Vendor Identifier' = (the IUT's Vendor_Identifier)

9.44.1.4 Device Object Instance Number is Configurable Even When MAC Address is Also Sent

Reason for change: Removed MS/TP Subordinate support and verify MAC Address unchanged.

Purpose: To verify the IUT can be configured with a Device object instance number even when a MAC address that does not match its own is also sent.

Test Concept: The ~~IUT sends a Who-Am-I while unconfigured.~~ TD assigns the IUT a Device object instance number (X) and also sends a MAC address (M1), which is different than the IUT's MAC address *(M0)*. The IUT will accept *the device identifier and may or may not save the new MAC address.* ~~both values, but not change its MAC address to M1.~~

Configuration Requirements: The IUT is not configured with a Device ID. The IUT is configured with a MAC address *(M0)*. ~~If the IUT cannot be configured in this way, this test shall be skipped. If the IUT cannot be configured with a MAC address that cannot be changed using You-Are, 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. MAKE (The IUT send a Who-Am-I)~~
~~2. RECEIVE~~
~~SOURCE = IUT~~
~~DESTINATION = TD | LOCAL BROADCAST | GLOBAL BROADCAST | REMOTE BROADCAST,~~
~~Who-Am-I Request,~~
~~'Vendor Identifier' = (the IUT's Vendor_Identifier),~~
~~'Model Name' = (the IUT's Model_Name),~~
~~'Serial Number' = (the IUT's Serial_Number)~~
~~3~~*1*. TRANSMIT
DESTINATION = *M0*~~IUT~~ | 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' = (Device, X),
'Device MAC Address' = (M1: a MAC address different *from M0*~~than that of the IUT~~)
~~4. IF (the IUT is not an MS/TP subordinate node)~~
~~5~~*2*. BEFORE Unconfirmed Response Fail Time
~~6~~*3*. RECEIVE
SOURCE = *M0 | (M1, if the IUT supports a changeable MAC Address)* ~~(if the IUT IUT ensure that the MAC address has not changed~~
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)
~~7. CHECK (the IUT's MAC address did not change)~~

© 2026 by BACnet International. All rights reserved.

9.44.1.5 MAC Address is Reconfigurable

Reason for change: Fixed misc MAC Address and Device instance references.

Purpose: To verify the IUT can be configured with a MAC address different *from* ~~than~~ the one it was configured with using the You-Are service~~, without having to be unconfigured first~~.

Test Concept: The ~~IUT is configured and does not need a MAC address configured to it.~~ TD assigns a different MAC address to the IUT using the You-Are service. The IUT changes its MAC address.

Configuration Requirements: The IUT is configured with a MAC address *(M0)*. ~~If the IUT cannot be configured with a MAC address that can be changed using You-Are, 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. READ (X = Device,4194303, ObjectIdentifier)
~~1. VERIFY (Device, IUT), Object_Type = DEVICE~~
2. TRANSMIT
DESTINATION = *M0*~~IUT~~ | 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' = (*X*~~the IUT's Device Object_Identifier~~),
'Device MAC Address' = (*M1,* a MAC address different than that of the IUT *(M0)*)
~~3. IF (the IUT is not an MS/TP subordinate node)~~
4*3*. BEFORE Unconfirmed Response Fail Time
~~5~~*4*. RECEIVE
SOURCE = *M1*~~IUT~~ -- ensure that the MAC address is what was set by TD
DESTINATION = TD | LOCAL BROADCAST | GLOBAL BROADCAST | REMOTE BROADCAST,
I-Am Request,
'Device Identifier' = (*X*~~the IUT's Device Object_Identifier~~),
'Max APDU Length Accepted' = (any valid value),
'Segmentation Supported' = (any valid value),
'Vendor Identifier' = (the IUT's Vendor_Identifier)
~~6. TRANSMIT~~
~~DESTINATION = IUT -- using the newly configured MAC address~~
~~ReadProperty-Request,~~
~~'Object Identifier' = (Device, IUT),~~
~~'Property Identifier' = Object_Type~~
~~7. RECEIVE~~
~~SOURCE = IUT -- ensure that the MAC address is what was set by TD~~
~~ReadProperty-Ack,~~
~~'Object Identifier' = (Device, IUT),~~
~~'Property Identifier' = Object_Type,~~
~~'Property Value' = DEVICE~~

9.44.1.6 Retains Configuration Through Restarts

Reason for change: Fixed the BEFORE values and now checks the MAC Address. Eliminated unnecessary steps.

Purpose: To verify the IUT will retain a configured Device object instance number and MAC address*, if configurable,* across a restart.

© 2026 by BACnet International. All rights reserved.

Test Concept: The IUT is configured with a Device object instance number (X) and MAC address (M1) using You-Are. The IUT is then restarted and the Device object instance number and MAC address are verified to be the same that were configured at the beginning of the test.

Configuration Requirements: ~~The IUT is unconfigured.~~ *The IUT is not configured with a MAC address or contains MAC address (M0). The IUT is not configured with a Device object instance number.*

Notes to Tester: ~~If the IUT's MAC address cannot be changed using You-Are, use the IUT's MAC address in place of M1.~~ The destination address used by TD shall be selected such that the IUT will receive the messages.

Test Steps:

1. TRANSMIT
   DESTINATION = *M0*~~IUT~~ | 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' = (Device, X),
   'Device MAC Address' = (M1~~, or absent if the IUT does not support a configurable MAC address~~)
2. *BEFORE Unconfirmed Response Fail Time*
~~2~~3. RECEIVE
   SOURCE = M0 | (M1, if the IUT supports a changeable MAC Address)
   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)
~~3~~4. MAKE (the IUT restart)
~~4~~5. WAIT (for the IUT to restart)
~~5~~6. TRANSMIT
   DESTINATION = IUT|LOCAL BROADCAST | GLOBAL BROADCAST | REMOTE BROADCAST,
   Who-Is Request,
   'Device Instance Range Low Limit' = 4194303,
   'Device Instance Range High Limit' = 4194303
~~6. WAIT Internal Processing Fail Time~~
7. *BEFORE Unconfirmed Response Fail Time*
8. *RECEIVE*
   SOURCE = M0 | (M1, if the IUT supports a changeable MAC Address)
   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)
~~7~~9. CHECK (the IUT did not send a Who-Am-I Request)
~~8. TRANSMIT~~
~~DESTINATION = IUT| LOCAL BROADCAST | GLOBAL BROADCAST | REMOTE BROADCAST,~~
~~Who-Is Request,~~
~~9. 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),~~

© 2026 by BACnet International. All rights reserved.

~~'Vendor Identifier' = (the IUT's Vendor_Identifier)~~
~~10. CHECK (the IUT's MAC address is M1 if it was changeable)~~

9.44.1.7 Unconfigurable by You-Are

Reason for change: Fixed the WAIT values.

Purpose: To verify the IUT can be unconfigured with the You-Are service.

Test Concept: The IUT is configured with a Device object instance number and MAC address. TD then sends a You-Are request with 'Device Identifier' set to ~~4149303~~*4194303*, and the ~~IUT~~ *IUT's Device object instance number* becomes unconfigured.

Configuration Requirements: The IUT is configured with both a Device object instance number and MAC address *(M0)*.

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 = ~~IUT~~*M0* | 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' = (Device, 4194303)
   'MAC Address' = (~~the IUT's MAC address~~*M0*, or absent)
2. *WAIT Unconfirmed Response Fail Time*
~~2~~*3*. CHECK (the IUT did not send any requests, except for an optional Who-Am-I if the MAC address cannot be unconfigured)
4. *TRANSMIT*
   DESTINATION = M0,
   ReadProperty-Request,
   'Object Identifier' = (Device, 4194303),
   'Property Identifier' = (any property in the object selected)
5. *CHECK (the IUT did not respond with a ReadProperty-ACK)*

9.44.2 Negative Tests

[Remove test 9.44.2.1]

9.44.2.2 Only Accepts Configuration When Received Parameters Match

Reason for change: Validates the device instance does not change.

Purpose: To verify the IUT will not ~~configure or~~ reconfigure itself when the parameters in a You-Are request do not match its vendor identifier, model name, and serial number.

Test Concept: The IUT is *configured* ~~unconfigured~~ and is sent a You-Are but with the wrong Vendor Identifier, Model Name, and Serial Number. The IUT does not accept the configuration and does not transmit an I-Am request indicating it has been configured.

Configuration Requirements: The IUT is configured with *Device object instance number and MAC address.* ~~needs configuration of either Device object instance number or MAC address, or both. This test shall be skipped if the IUT is an MS/TP subordinate node.~~

© 2026 by BACnet International. All rights reserved.

Notes to Tester: ~~If the IUT only supports configuration of either Device object instance number or MAC address but not both, TD shall use the IUT's Device Identifier or MAC address, whichever is configured, when sending You-Are requests.~~ The destination address used by TD shall be selected such that the IUT will receive the messages.

Test Steps:

1. READ (X = Device,4194303, ObjectIdentifier)
~~1~~*2*. TRANSMIT
DESTINATION = IUT | 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' = (any value other than the IUT's Serial_Number),
'Device Identifier' = (any valid Device Identifier *not equal to X*),
'Device MAC Address' = (~~any valid MAC address, or~~ absent)
~~2~~*3*. WAIT Unconfirmed Response Fail Time
~~3. CHECK (the IUT did not transmit an I-Am-Request)~~
4. TRANSMIT
DESTINATION = IUT | LOCAL BROADCAST | GLOBAL BROADCAST | REMOTE BROADCAST,
You-Are Request,
'Vendor Identifier' = (the IUT's Vendor_Identifier),
'Model Name' = (any value other than the IUT's Model_Name),
'Serial Number' = (IUT's Serial_Number),
'Device Identifier' = (any valid Device Identifier *not equal to X*),
'Device MAC Address' = (~~any valid MAC address, or~~ absent)
5. WAIT Unconfirmed Response Fail Time
~~6. CHECK (the IUT did not transmit an I-Am-Request)~~
~~7~~*6*. TRANSMIT
DESTINATION = IUT | LOCAL BROADCAST | GLOBAL BROADCAST | REMOTE BROADCAST,
You-Are Request,
'Vendor Identifier' = (any value other than the IUT's Vendor_Identifier),
'Model Name' = (the IUT's Model_Name),
'Serial Number' = (the IUT's Serial_Number),
'Device Identifier' = (any valid Device Identifier *not equal to X*),
'Device MAC Address' = (~~any valid MAC address, or~~ absent)
~~8~~*7*. WAIT Unconfirmed Response Fail Time
~~9. CHECK (the IUT did not transmit an I-Am-Request)~~
8. VERIFY (Device, 4194303, ObjectIdentifier = X)

© 2026 by BACnet International. All rights reserved.

10. NETWORK LAYER PROTOCOL TESTS

10.2 Router Functionality Tests

10.2.2 Processing Network Layer Messages

10.2.2.4.3 Receiving Messages for a Busy Router

Reason for Change: Corrects problems with test in 135.1-2025 in step 3.

Purpose: To verify that the IUT rejects a message destined for a busy router.

Test Steps:

1. TRANSMIT PORT B,
   DESTINATION = LOCAL BROADCAST,
   SOURCE = R2-3,
   Router-Busy-To-Network,
   Network Numbers = 3
2. RECEIVE PORT A,
   DESTINATION = LOCAL BROADCAST,
   SOURCE = IUT,
   Router-Busy-To-Network,
   Network Numbers = 3
3. TRANSMIT PORT A,
~~10.2.2.4.3.1.1.1.1.~~ DA = IUT,
   SOURCE = D1A,
   DNET = 3,
   DADR = D3D,
   Hop Count = 255,
   ReadProperty-Request,
   'Object Identifier' = (any BACnet standard object),
   'Property Identifier' = (any required property of the specified object)
4. RECEIVE PORT A,
   DESTINATION = D1A,
   SOURCE = IUT,
   Reject-Message-To-Network,
   Reject Reason = 2 (router busy),
   DNET = 3

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 ~~10s +~~ **Internal Processing Fail Time**

© 2026 by BACnet International. All rights reserved.

3. RECEIVE Network-Number-Is,
Network Number = *N1*~~(the configured value)~~,
Configured =(any valid value)

10.7 Route Binding Tests

10.7.3 Router Binding via Who-Is-Router-To-Network

Reason for Change: Modify test to allow for the case where IUT is a router that does not support initiating confirmed requests. Fix test to allow for unconfirmed only devices.

Purpose: To verify that the IUT can ~~initiate~~*send* requests to a remote network after the IUT uses the Who-Is-Router-To-Network Network Layer service to discover the MAC address of the router to that remote network.

Test Concept: The IUT broadcasts a Who-Is-Router-To-Network request to discover the router to the desired network. The ~~TD~~*IUT* transmits a request to a device on the remote network without performing any further form of dynamic router binding. ~~If the IUT does not support Who-Is-Router-To-Network router binding, then this test shall be omitted. If the IUT cannot initiate a ReadProperty request, then another confirmed service can be substituted.~~ *If the IUT is a router that does not support initiating service requests then it may forward a confirmed service request from device D4A instead.* The IUT may use either the general query or specific network number query form of the Who-Is-Router-To-Network service.

Note that Clause 6.5.3 specifically mentions router binding via Who-Is-Router-To-Network and does not mention router binding by lurking and noting unsolicited I-Am-Router-To-Network messages.

Test Steps:

1. MAKE (IUT transmit Who-Is-Router-To-Network to discover the router to DNET2)
2. RECEIVE
DA = BROADCAST,
SA = IUT,
Who-Is-Router-To-Network,
| (DA = BROADCAST,
SA = IUT,
Who-Is-Router-To-Network,
DNET =DNET2)
3. TRANSMIT
DESTINATION = BROADCAST,
SOURCE = TD,
I-Am-Router-To-Network,
Network Numbers = DNET2
4. IF (the IUT can initiate a request) THEN {
~~4~~*5*. MAKE (IUT transmit a ~~ReadProperty~~ request to the D2A device on the remote network)
~~5~~*6*. RECEIVE
DA = TD,
SA = IUT,
DNET = DNET2,
DADR= D2A,
Hop Count = 255,
BACnet-Confirmed-Request-PDU,
'Service Choice' = ~~ReadProperty-Request,~~ *Any,*
~~'Object Identifier' = (O1, any BACnet standard object in D2A),~~
~~'Property Identifier' = (P1, any required property of the specified object)~~
|*DA = TD,*
SA = IUT,

© 2026 by BACnet International. All rights reserved.

DNET = DNET2,
DADR= D2A,
Hop Count = 255,
BACnet-Unconfirmed-Request-PDU,
'Service Choice' = Any,

|DA = TD,
SA = IUT,
DNET = DNET2,
DADR= Absent,
Hop Count = 255,
BACnet-Unconfirmed-Request-PDU,
'Service Choice' = Any,

~~6. TRANSMIT~~
~~DA = IUT,~~
~~SA = TD,~~
~~SNET = DNET2,~~
~~SADR = D2A,~~
~~BACnet-ComplexACK-PDU,~~
~~'Service ACK Choice' = ReadProperty-ACK,~~
~~'Object Identifier' = O1,~~
~~'Property Identifier' = P1,~~
~~'Property Value' = (any valid value)~~

7. IF (if a confirmed Request was requested) THEN
8. TRANSMIT
DA = IUT,
SA = TD,
SNET = DNET2,
SADR = D2A,
(A valid BACnet Response)
}
ELSE { -- (IUT is a router that does not support initiating ~~confirmed~~ service requests)
9. TRANSMIT
DA = IUT,
SA = D4A,
DNET = DNET2,
DADR = D2A,
Hop Count = 255,
BACnet-Confirmed-Request-PDU,
'Service Choice' = ReadProperty-Request,
'Object Identifier' = (O1, any BACnet standard object in D2A),
'Property Identifier' = (P1, any required property of the specified object)
10. RECEIVE
DA = TD,
SA = IUT,
DNET = DNET2,
DADR = D2A,
SNET = DNET4
SADR = D4A
Hop Count = (any integer x: 0 < x < 255),
BACnet-Confirmed-Request-PDU,
'Service Choice' = ReadProperty-Request,
'Object Identifier' = O1
'Property Identifier' = P1
11. TRANSMIT
DA = IUT,
SA = TD,

© 2026 by BACnet International. All rights reserved.

DNET = DNET4,
DADR = D4A,
SNET = DNET2,
SADR = D2A
Hop Count = 254,
BACnet-Confirmed-Request-PDU,
'Service Choice' = ReadProperty-ACK,
'Object Identifier' = O1
'Property Identifier' = P1
'Property Value' = (any valid value)
}

10.8 Virtual Routing Functionality Tests

10.8.4 Routing of Broadcast APDUs to Virtual Devices

10.8.4.1 Broadcasts that Should Be Ignored

Reason for Change: Removed optional tests.

Purpose: To verify that the IUT will not route APDUs which are locally broadcast on the directly connected physical BACnet network or remotely broadcast to a network that is reachable through the same port the message was received from.

Test Concept: *Initiate a local broadcast Who-Is message to any virtual device, VD1A, on the virtual network and check the IUT does not initiate an I-Am message for VD1A. Repeat the test with a directed Who-Is from a remote network and again with a local broadcast Who-Is message destined for a remote network.* ~~In order to verify that a broadcast message is not routed, we need to look for some indication that the message was routed. Two commonly supported services that can be used for this test are Who-Is and Who-Has, but complications may arise because a device is allowed to transmit the expected responses (I-Am and I-Have, respectively) at any time. There are a few other services that may also be used, but special device configuration will most likely be required. The tester shall choose one of the following test options, with the requirement that the option chosen must use services supported by at least one of the IUT's virtual devices.~~

~~This test shall be skipped if the IUT's virtual devices are not capable of executing any application services that may be broadcast.~~

~~A. Who-Is Option~~

~~This test option is an alternative for IUTs having a virtual device that supports the execution of the Who-Is service. This test shall be run after all tests for the execution of the Who-Is service have been run with a passing result. One virtual device that supports the execution of the Who-Is service shall be selected for this test.~~

Test Steps:

1. TRANSMIT
   DA = LOCAL BROADCAST,
   SA = TD,
   Who-Is-Request,
   'Device Instance Range Low Limit' = (~~the Device object instance number of the virtual device~~*VD1A*),
   'Device Instance Range High Limit' = (~~the Device object instance number of the virtual device~~*VD1A*)
2. WAIT **Internal Processing Fail Time**
3. CHECK (verify that the IUT does not transmit an I-Am message *for VD1A*~~from the virtual device selected in step 1~~)
4. TRANSMIT
   DA = LOCAL BROADCAST,
   SA = R2-3,
   SNET = 3,

© 2026 by BACnet International. All rights reserved.

SADR = D3D,
Who-Is-Request,
'Device Instance Range Low Limit' = (~~the Device object instance number of the virtual device~~*VD1A*)
'Device Instance Range High Limit' = (~~the Device object instance number of the virtual device~~*VD1A*)

5. WAIT **Internal Processing Fail Time**
6. CHECK (verify that the IUT does not transmit an I-Am message *for VD1A*~~from the virtual device selected in step 4~~)
7. TRANSMIT
   DA = LOCAL BROADCAST,
   SA = TD,
   DNET = 3,
   DLEN = 0,
   Hop Count = 255,
   Who-Is-Request,
   'Device Instance Range Low Limit' = (~~the Device object instance number of the virtual device~~*VD1A*)
   'Device Instance Range High Limit' = (~~the Device object instance number of the virtual device~~*VD1A*)
8. WAIT **Internal Processing Fail Time**
9. CHECK (verify that the IUT does not transmit an I-Am message *for VD1A*~~from the virtual device selected in step 7~~)

~~Notes to Tester: I-Am messages may be sent at any time by any device. If one or both of the CHECK steps fail, then repeat the test until you can determine with confidence whether the I-Am messages are in response to the Who-Is (a failing result) or are being transmitted for some other reason (a passing result).~~

~~B. Who-Has Option~~

~~This test option is an alternative for IUTs that have a virtual device that supports the execution of the Who-Has service. This test shall be run after all tests for the execution of the Who-Has service have been run with a passing result. One virtual device that supports the execution of the Who-Has service shall be selected for this test.~~

~~Test Steps:~~

~~1. TRANSMIT~~
~~DA = LOCAL BROADCAST,~~
~~SA = TD,~~
~~Who-Has-Request,~~
~~'Device Instance Range Low Limit' = (the Device object instance number of the virtual device),~~
~~'Device Instance Range High Limit' = (the Device object instance number of the virtual device),~~
~~'Object Identifier' = (any object identifier of an object in the virtual device)~~
~~2. WAIT **Internal Processing Fail Time**~~
~~3. CHECK (verify that the IUT does not transmit an I-Have message from the virtual device selected in step 1 that contains the object identifier used in step 1)~~
~~4. TRANSMIT~~
~~DA = LOCAL BROADCAST,~~
~~SA = R2-3,~~
~~SNET = 3,~~
~~SADR = D3D,~~
~~Who-Has-Request,~~
~~'Device Instance Range Low Limit' = (the Device object instance number of the virtual device),~~
~~'Device Instance Range High Limit' = (the Device object instance number of the virtual device),~~
~~'Object Identifier' = (any object identifier of an object in the virtual device)~~
~~5. WAIT **Internal Processing Fail Time**~~
~~6. CHECK (verify that the IUT does not transmit an I-Have message from the virtual device selected in step 4 that contains the object identifier used in step 4)~~
~~7. TRANSMIT~~
~~DA = LOCAL BROADCAST,~~
~~SA = TD,~~
~~DNET = 3,~~

© 2026 by BACnet International. All rights reserved.

~~DLEN = 0,~~
~~Hop Count = 255,~~
~~Who-Has-Request,~~
~~'Device Instance Range Low Limit' = (the Device object instance number of the virtual device),~~
~~'Device Instance Range High Limit' = (the Device object instance number of the virtual device),~~
~~'Object Identifier' = (any object identifier of an object in the virtual device)~~
~~8. WAIT **Internal Processing Fail Time**~~
~~9. CHECK (verify that the IUT does not transmit an I-Have message from the virtual device selected in step 4 that contains the object identifier used in step 7)~~

~~Notes to Tester: I-Have messages may be sent at any time by any device. If one or both of the CHECK steps fail, then repeat the test until you can determine with confidence whether the I-Have messages are in response to the Who-Has (a failing result) or are being transmitted for some other reason (a passing result).~~

~~C. Generic Test Option~~

~~This test option is intended for IUTs whose virtual devices do not support the execution of the Who-Has or Who-Is services. This option uses a service that may be broadcast and that is supported by one of the IUT's virtual devices. This test shall be run after all tests for the execution of the particular service have been run with a passing result.~~

~~Configuration requirements: Configure the IUT so that the receipt of a particular service request by a particular virtual device will cause some indication that is visible to the tester. Verify that the indication occurs if the service request is sent directly to the virtual device or remote broadcast on Network 1.~~

~~Test Steps:~~

~~1. TRANSMIT~~
~~DA = LOCAL-BROADCAST,~~
~~SA = TD,~~
~~BACnet-Unconfirmed-Request-PDU~~
~~2. WAIT **Internal Processing Fail Time**~~
~~3. CHECK (verify that the indication does not occur)~~
~~4. TRANSMIT~~
~~DA = LOCAL-BROADCAST,~~
~~SA = R2-3,~~
~~SNET = 3,~~
~~SADR = D3D,~~
~~BACnet-Unconfirmed-Service-Request~~
~~5. WAIT **Internal Processing Fail Time**~~
~~6. CHECK (verify that the indication does not occur)~~
~~7. TRANSMIT~~
~~DA = LOCAL-BROADCAST,~~
~~SA = TD,~~
~~DNET = 3,~~
~~DLEN = 0,~~
~~Hop Count = 255,~~
~~BACnet-Unconfirmed-Service-Request~~
~~8. WAIT **Internal Processing Fail Time**~~
~~9. CHECK (verify that the indication does not occur)~~

10.8.4.2 Route Global Broadcast from a Local Device to Virtual Devices

Reason for Change: Removed optional tests.

Purpose: To verify that the IUT properly forwards global broadcast messages originating on a local network to its virtual devices. ~~This test shall be skipped if the IUT's virtual devices are not capable of executing any application services that may be broadcast.~~

© 2026 by BACnet International. All rights reserved.

Test Concept: *Initiate a global broadcast Who-Has message to any virtual device, VD1A, on the virtual network and check the IUT initiate an I-Have message for VD1A.* ~~A broadcast message of a type that the virtual device will execute is sent to the IUT. It is then verified that a correct response is generated.~~

~~The tester selects one of the following test options, depending on which services are supported by the virtual devices in the IUT.~~

~~A. Who-Is Test Option~~

~~Select one virtual device in the IUT that supports the execution of Who-Is requests.~~

~~Test Steps:~~

~~1. TRANSMIT~~
~~DA = LOCAL BROADCAST,~~
~~SA = D2C,~~
~~DNET = GLOBAL BROADCAST,~~
~~DLEN = 0,~~
~~Hop Count = 255,~~
~~Who-Is-Request,~~
~~'Device Instance Range Low Limit' = (the Device object instance number of the virtual device),~~
~~'Device Instance Range High Limit' = (the Device object instance number of the virtual device)~~
~~2. RECEIVE~~
~~DA = LOCAL BROADCAST,~~
~~SA = IUT,~~
~~DNET = GLOBAL BROADCAST,~~
~~DLEN = 0,~~
~~SNET = 1,~~
~~SADR = (the MAC address of the virtual device),~~
~~Hop Count = (any integer x: 1 < x < 255),~~
~~I-Am-Request,~~
~~'I Am Device Identifier' = (the Device object instance number of the virtual device),~~
~~'Max APDU Length Accepted' = (any valid value),~~
~~'Segmentation Supported' = (any valid value),~~
~~'Vendor Identifier' = (any valid value)~~
~~(DA = LOCAL BROADCAST | D2C,~~
~~SA = IUT,~~
~~SNET = 1,~~
~~SADR = (the MAC address of the virtual device),~~
~~I-Am-Request,~~
~~'I Am Device Identifier' = (the Device object instance number of the virtual device),~~
~~'Max APDU Length Accepted' = (any valid value),~~
~~'Segmentation Supported' = (any valid value),~~
~~'Vendor Identifier' = (any valid value))~~

~~B. Who-Has Test Option~~

~~Select one virtual device in the IUT that supports the execution of Who-Has requests.~~

Test Steps:

1. TRANSMIT
   DA = LOCAL BROADCAST,
   SA = TD,
   DNET = GLOBAL BROADCAST,

© 2026 by BACnet International. All rights reserved.

DLEN = 0,
Hop Count = 255,
Who-Has-Request,
'Device Instance Range Low Limit' = (~~the Device object instance number of the virtual device~~*VD1A*)
'Device Instance Range High Limit' = (~~the Device object instance number of the virtual device~~*VD1A*)
'Object Identifier' = (*O1,* any object identifier of an object in the virtual device)

2. RECEIVE
DA = LOCAL BROADCAST,
SA = IUT,
DNET = GLOBAL BROADCAST,
DLEN = 0,
SNET = 1,
SADR = (the MAC address of ~~the virtual device~~*VD1A*),
Hop Count = (any integer x: 1 < x < 255),
I-Have-Request,
'Device Identifier' = (~~the Device object instance number of the virtual device~~*VD1A*),
'Object Identifier' = (~~the object identifier specified in step 1~~*O1*),
~~'Object Name' = (any valid value)~~
'Object Name' = (any valid value) |
(DA = LOCAL BROADCAST,
SA = IUT,
SNET = 1,
SADR = (the MAC address of ~~the virtual device~~*VD1A*),
I-Have-Request,
'Device Identifier' = (~~the Device object instance number of the virtual device~~*VD1A*),
'Object Identifier' = (~~the object identifier specified in step 1~~*O1*),
'Object Name' = (any valid value))

~~C. Generic Test Option~~

~~This test option is intended for IUTs whose virtual devices do not support the execution of the Who-Has or Who-Is services. This option uses a service that may be broadcast and that is supported by one of the IUT's virtual devices. This test shall be run after all tests for the execution of the particular service have been run with a passing result.~~

~~Configuration requirements: Configure the IUT so that the receipt of a particular service request by a particular virtual device will cause some indication that is visible to the tester.~~

~~Test Steps:~~

~~1. TRANSMIT~~
~~DA = LOCAL BROADCAST,~~
~~SA = TD,~~
~~DNET = GLOBAL BROADCAST,~~
~~DLEN = 0,~~
~~Hop Count = 255,~~
~~BACnet-Unconfirmed-Request-PDU~~
~~2. CHECK (verify that the indication occurs)~~

10.8.4.3 Route Global Broadcast from a Remote Device to Virtual Devices

Reason for Change: Removed optional tests.

Purpose: To verify that the IUT properly forwards global broadcast messages that originate on a remote network to its virtual devices. ~~This test shall be skipped if the IUT's virtual devices are not capable of executing any application services that may be broadcast.~~

© 2026 by BACnet International. All rights reserved.

Test Concept: Initiate a global broadcast Who-Is message from a remote device to any virtual device, VD1A, on the virtual network and check the IUT initiate an I-Am message for VD1A.
~~The tester shall select one of the following test options, depending on what services are supported by the virtual devices in the IUT.~~

~~A. Who-Is Test Option~~

~~Select one virtual device in the IUT that supports the execution of Who-Is requests.~~

Test Steps:

1. TRANSMIT
   DA = LOCAL BROADCAST,
   SA = R2-3,
   DNET = GLOBAL BROADCAST,
   DLEN = 0,
   SNET = 3,
   SADR = D3D,
   Hop Count = 254,
   Who-Is-Request,
   'Device Instance Range Low Limit' = (~~the Device object instance number of the virtual device~~*VD1A*)
   'Device Instance Range High Limit' = (~~the Device object instance number of the virtual device~~*VD1A*)
2. RECEIVE
   DA = LOCAL BROADCAST,
   SA = IUT,
   DNET = GLOBAL BROADCAST,
   DLEN = 0,
   SNET = 1,
   SADR = (the MAC address of ~~the virtual device~~*VD1A*),
   Hop Count = (any integer x: 1 < x < 255),
   I-Am-Request,
   'I Am Device Identifier' = (~~the Device object instance number of the virtual device~~*VD1A*),
   'Max APDU Length Accepted' = (any valid value),
   'Segmentation Supported' = (any valid value),
   ~~'Vendor Identifier' = (any valid value)~~
   'Vendor Identifier' = (any valid value) |
   (DA = R2-3,
   SA = IUT,
   DNET = 3,
   DLEN = 0,
   SNET = 1,
   SADR = (the MAC address of ~~the virtual device~~*VD1A*),
   Hop Count = (any integer x: 1 < x < 255),
   I-Am-Request,
   'I Am Device Identifier' = (~~the Device object instance number of the virtual device~~*VD1A*),
   'Max APDU Length Accepted' = (any valid value),
   'Segmentation Supported' = (any valid value),
   ~~'Vendor Identifier' = (any valid value))~~
   'Vendor Identifier' = (any valid value) |
   (DA = R2-3,
   SA = IUT,
   DNET = 3,
   DADR = (the MAC address of D3D),
   SNET = 1,
   SADR = (the MAC address of ~~the virtual device~~*VD1A*),
   Hop Count = (any integer x: 1 < x < 255),

© 2026 by BACnet International. All rights reserved.

I-Am-Request,
'I Am Device Identifier' = (~~the Device object instance number of the virtual device~~*VD1A*),
'Max APDU Length Accepted' = (any valid value),
'Segmentation Supported' = (any valid value),
'Vendor Identifier' = (any valid value))

~~B. Who-Has Test Option~~

~~Select one virtual device in the IUT that supports the execution of Who-Has requests.~~

~~Test Steps:~~

~~1. TRANSMIT~~
~~DA = LOCAL-BROADCAST,~~
~~SA = R2-3,~~
~~DNET = GLOBAL-BROADCAST,~~
~~DLEN = 0,~~
~~SNET = 3,~~
~~SADR = D3D,~~
~~Hop Count = 254,~~
~~Who-Has-Request,~~
~~'Device Instance Range Low Limit' = (the Device object instance number of the virtual device),~~
~~'Device Instance Range High Limit' = (the Device object instance number of the virtual device),~~
~~'Object Identifier' = (any object identifier of an object in the virtual device)~~
~~2. RECEIVE~~
~~DA = LOCAL-BROADCAST,~~
~~SA = IUT,~~
~~DNET = GLOBAL-BROADCAST,~~
~~DLEN = 0,~~
~~SNET = 1,~~
~~SADR = (the MAC address of the virtual device),~~
~~Hop Count = (any integer x: 1 < x < 255),~~
~~I-Have-Request,~~
~~'Device Identifier' = (the Device object instance number of the virtual device),~~
~~'Object Identifier' = (the object identifier specified in step 1),~~
~~'Object Name' = (any valid value)~~
~~(DA = R2-3,~~
~~SA = IUT,~~
~~DNET = 3,~~
~~DLEN = 0,~~
~~SNET = 1,~~
~~SADR = (the MAC address of the virtual device),~~
~~Hop Count = (any integer x: 1 < x < 255),~~
~~I-Have-Request,~~
~~'Device Identifier' = (the Device object instance number of the virtual device),~~
~~'Object Identifier' = (the object identifier specified in step 1),~~
~~'Object Name' = (any valid value))~~

~~C. Generic Test Option~~

~~This test option is intended for IUTs whose virtual devices do not support the execution of the Who-Has or Who-Is services. This option uses a service that may be broadcast and that is supported by one of the IUT's virtual devices. This test shall be run after all tests for the execution of the particular service have been run with a passing result.~~

~~Configuration requirements: Configure the IUT so that the receipt of a particular service request by a particular virtual device will cause some indication that is visible to the tester.~~

© 2026 by BACnet International. All rights reserved.

~~Test Steps:~~

~~1. TRANSMIT~~
~~DA = LOCAL BROADCAST,~~
~~SA = R2_3,~~
~~DNET = GLOBAL BROADCAST,~~
~~DLEN = 0,~~
~~SNET = 3,~~
~~SADR = D3D,~~
~~Hop Count = 254,~~
~~BACnet-Unconfirmed-Request-PDU~~
~~2. CHECK (verify that the indication occurs)~~

10.8.4.4 Route Remote Broadcast from a Local Device to Virtual Devices

Reason for Change: Copied 10.8.4.2 Step 2 instead of referencing it.

Purpose: To verify that the IUT properly forwards remote broadcast messages originating on a local network to its virtual devices.

Test Concept: *Initiate a remote broadcast Who-Is message from a local device to any virtual device, VD1A, on the virtual network and check the IUT initiates an I-Am message for VD1A.*~~Execute test 10.8.4.2 (Route Global Broadcast from a Local Device to Virtual Devices) modified as follows. Instead of transmitting a global broadcast message, the TD shall transmit a remote broadcast message from a device on Network 2 directed to Network 1:~~

Test Steps:

1. TRANSMIT
   DA = LOCAL BROADCAST,
   SA = TD,
   DNET = 1,
   DLEN = 0,
   Hop Count = 255,
   ~~BACnet-Unconfirmed-Request-PDU~~
~~2. Continue with Step 2 from test 10.8.4.2~~
   Who-Is-Request,
   'Device Instance Range Low Limit' = (VD1A),
   'Device Instance Range High Limit' = (VD1A)
2. *RECEIVE*
   DA = LOCAL BROADCAST,
   SA = IUT,
   DNET = GLOBAL BROADCAST,
   DLEN = 0,
   SNET = 1,
   SADR = (the MAC address of VD1A),
   Hop Count = (any integer x: 1 < x < 255),
   I-Am-Request,
   'I Am Device Identifier' = (VD1A),
   'Max APDU Length Accepted' = (any valid value),
   'Segmentation Supported' = (any valid value),
   'Vendor Identifier' = (any valid value) |
   (DA = LOCAL BROADCAST,
   SA = IUT,
   SNET = 1,
   SADR = (the MAC address of VD1A),
   I-Am-Request,

© 2026 by BACnet International. All rights reserved.

'I Am Device Identifier' = (VD1A),
'Max APDU Length Accepted' = (any valid value),
'Segmentation Supported' = (any valid value),
'Vendor Identifier' = (any valid value))

~~This test shall be skipped if the IUT's virtual devices are not capable of executing any application services that may be broadcast.~~

10.8.4.5 Route Remote Broadcast from a Remote Device to Virtual Devices

Reason for Change: Copied 10.8.4.3 Step 2 instead of referencing it.

Purpose: To verify that the IUT properly forwards remote broadcast messages that originate on a remote network to its virtual devices.

Test Concept: *Initiate a remote broadcast Who-Is message from a remote device to any virtual device, VD1A, on the virtual network and check the IUT initiates an I-Am message for VD1A.*~~Execute test 10.8.4.3 (Route Global Broadcast from a Remote Device to Virtual Devices) modified as follows. Instead of transmitting a global broadcast message, the TD shall transmit a remote broadcast message from a device on Network 3 directed to Network 1:~~

Test Steps:

1. TRANSMIT
   DA = IUT,
   SA = R2-3,
   DNET = 1,
   DLEN = 0,
   SNET = 3,
   SADR = D3D,
   Hop Count = 254,
   ~~BACnet-Unconfirmed-Request-PDU~~
~~2. Continue with Step 2 from test 10.8.4.3~~
   Who-Is-Request,
   'Device Instance Range Low Limit' = (VD1A)
   'Device Instance Range High Limit' = (VD1A)
2. *RECEIVE*
   DA = LOCAL BROADCAST,
   SA = IUT,
   DNET = GLOBAL BROADCAST,
   DLEN = 0,
   SNET = 1,
   SADR = (the MAC address of VD1A),
   Hop Count = (any integer x: 1 < x < 255),
   I-Am-Request,
   'I Am Device Identifier' = (VD1A),
   'Max APDU Length Accepted' = (any valid value),
   'Segmentation Supported' = (any valid value),
   'Vendor Identifier' = (any valid value) |
   (DA = R2-3,
   SA = IUT,
   DNET = 3,
   DLEN = 0,
   SNET = 1,
   SADR = (the MAC address of VD1A),
   Hop Count = (any integer x: 1 < x < 255),
   I-Am-Request,
   'I Am Device Identifier' = (VD1A),

© 2026 by BACnet International. All rights reserved.

'Max APDU Length Accepted' = (any valid value),
'Segmentation Supported' = (any valid value),
'Vendor Identifier' = (any valid value) |
(DA = R2-3,
SA = IUT,
DNET = 3,
DADR = (the MAC address of D3D),
SNET = 1,
SADR = (the MAC address of VD1A),
Hop Count = (any integer x: 1 < x < 255),
I-Am-Request,
'I Am Device Identifier' = (VD1A),
'Max APDU Length Accepted' = (any valid value),
'Segmentation Supported' = (any valid value),
'Vendor Identifier' = (any valid value))

~~This test shall be skipped if the IUT's virtual devices are not capable of executing any application services that may be broadcast.~~

10.8.4.6 Route Global Broadcast Message from a Virtual Device

Reason for Change: Fixed incorrect broadcast type.

Purpose: To verify that the IUT can route a global broadcast message from one of its virtual devices to the local (physical) network.

Test Concept: Make one of the virtual devices generate a *global*~~remote~~ broadcast, and verify that it is correctly formulated. This test shall be skipped if none of the IUT's virtual devices can transmit a global broadcast message.

Test Steps:

1. MAKE (the virtual device generate a *global*~~remote~~ broadcast message)
2. RECEIVE
   DA = LOCAL BROADCAST,
   SA = IUT,
   DNET = GLOBAL BROADCAST,
   DLEN = 0,
   SNET = 1,
   SADR = (MAC address of a virtual device on Network 1),
   Hop Count = (any integer x: 1 < x < 255),
   BACnet-Unconfirmed-Request-PDU

10.8.4.8 Route Remote Broadcast Message from a Virtual Device to a Remote Network

Reason for Change: Removed unnecessary Note to Tester.

Purpose: To verify that the IUT can route a remote broadcast message from a virtual device to a remote network.

Test Concept: The IUT shall be configured such that its routing table only contains entries for the directly connected networks (physical and virtual). One of the virtual devices is made to send a remote broadcast message to Network 3.

Configuration Requirements: The IUT shall be configured such that its routing table only contains entries for the directly connected networks (physical and virtual). This test shall be skipped if none of the IUT's virtual devices can issue a remote broadcast message.

Test Steps:

© 2026 by BACnet International. All rights reserved.

1. MAKE (the virtual device generate a remote broadcast message to ~~the~~ network 3)
2. RECEIVE
   DA = R2-3,
   SA = IUT,
   DNET = 3,
   DLEN = 0,
   SNET = 1,
   SADR = (MAC address of a virtual device on Network 1),
   Hop Count = (any integer x: 1 < x < 255),
~~BACnet-Unconfirmed-Request-PDU~~
~~| (DA = LOCAL_BROADCAST,~~
   BACnet-Unconfirmed-Request-PDU |
   (DA = LOCAL_BROADCAST,
   SA = IUT,
   DNET = 3,
   DLEN = 0,
   SNET = 1,
   SADR = (MAC address of a virtual device on Network 1),
   Hop Count = (any integer x: 1 < x < 255),
   BACnet-Unconfirmed-Request-PDU)

~~Notes to Tester: During the test, the TD shall answer any requests that the IUT generates while attempting to locate the route to network 3.~~

© 2026 by BACnet International. All rights reserved.

12. DATA LINK LAYER PROTOCOLS TESTS

12.1 MS/TP State Machine Tests

12.1.3 MS/TP Data Link Layer Tests (Alternate)

12.1.3.10 Max_Manager Test

Reason for Change: Updated read-only verification to only run if prior to PR 17.

Purpose: Verify that Max_Manager *property in the Device Object* is writable, or that it is 127.

Configuration Requirements: The IUT shall be configured with an MS/TP MAC address less than 127.

Test Steps:

1. VERIFY Max_Manager = (any valid value, V1)
2. IF Max_Manager is writable THEN
3. WRITE Max_Manager = (any other valid value, ~~with V2~~ greater than or equal to the IUT's address)
~~3~~4. ELSE *IF (Protocol_Revision < 17) THEN*
5. VERIFY Max_Manager = 127

12.1.3.11 Max_Info_Frames Test

Reason for Change: Updated read-only verification to only run if prior to PR 17.

Purpose: Verify that Max_Info_Frames *property in the Device Object* is *writable,* configurable, or that it is 1.

Test Steps:

1. VERIFY Max_Info_Frames = (any valid value)
2. IF Max_Info_Frames is writable THEN
3. WRITE Max_Info_Frames = (any valid value, V)
4. VERIFY Max_Info_Frames = V
5. ELSE IF Max_Info_Frames is configurable THEN
6. MAKE (Max_Info_Frames equal any valid value, V)
7. VERIFY Max_Info_Frames = V
8. ELSE *IF (Protocol_Revision < 17) THEN*
9. VERIFY Max_Info_Frames = 1

12.1.3.X1 Max_Manager Test for Network Port Object

Reason for Change: New test for Network Port Object properties.

Purpose: Verify writing to the Max_Manager property in the Device object and the corresponding Network Port object are changed when the other is written. If neither are writable, the values must be 127.

Test Concept: Write the Device object (D1), Max_Manager property and check the corresponding Network Port object (NP1) is updated and the change does not require activation via the Network Port object. Write NP1, Max_Manager and check the corresponding Device object is updated.

Configuration Requirements: NP1 is the Network Port object with the lowest object instance whose Network_Type is MSTP and whose Out_Of_Service is FALSE.

Test Steps:

© 2026 by BACnet International. All rights reserved.

-- verify write via Device object
1. IF (D1, Max_Manager is writable) THEN {
2. READ V1 = D1, Max_Manager
3. WRITE D1, Max_Manager = (any valid value V2, other than V1 greater than or equal to the IUT's address)
4. WAIT (Internal Processing Fail Time)
5. VERIFY (NP1, Max_Manager = V2)
}

-- verify write via Network Port object
6. IF (NP1, Max_Manager is writable) THEN {
7. READ V1 = NP1, Max_Manager
8. VERIFY (D1, Max_Manager = V1)
9. WRITE NP1, Max_Manager = (any valid value V2, other than V1 greater than or equal to the IUT's address)
10. VERIFY NP1, Changes_Pending = TRUE
11. TRANSMIT ReinitializeDevice-Request
'Reinitialized State of Device' = WARMSTART | ACTIVATE_CHANGES
'Password' = (any valid password)
12. RECEIVE BACnet-SimpleACK-PDU
13. WAIT Activate Changes Fail Time
14. VERIFY NP1, Changes_Pending = FALSE
15. VERIFY (D1, Max_Manager = V2)
}
16. IF (neither D1, Max_Manager nor NP1, Max_Manager is writable) THEN {
17. VERIFY(NP1, Max_Manager = 127)
18. VERIFY(D1, Max_Manager = 127)
}

12.1.3.X2 Max_Info_Frames Test for Network Port Objects

Reason For Change: Added new for Network Port Object property testing.

Purpose: Verify modifying the Max_Info_Frames property in the Device object and the corresponding Network Port object (NP1) are changed when the other is written. If neither are writable, verify the values are equal to 1.

Test Concept: Modify the Device object (D1), Max_Info_Frames property and check the corresponding Network Port object (NP1) is updated and the change does not require activation via the Network Port object. Write NP1, Max_Info_Frames and check the corresponding Device object is updated.

Configuration Requirements: NP1 is the Network Port object with the lowest object instance whose Network_Type is MSTP and whose Out_Of_Service is FALSE.

Test Steps:

-- verify modify via Device object
1. READ V1 = D1, Max_Info_Frames
2. IF (D1, Max_Info_Frames is writable) THEN
3. WRITE D1, Max_Info_Frames = (V2, any valid value other than V1)
4. WAIT (Internal Processing Fail Time)
5. VERIFY (NP1, Max_Info_Frames = V2)
6. ELSE IF (D1, Max_Info_Frames is configurable) THEN {
7. MAKE (D1, Max_Info_Frames equal V2, any valid value other than V1)
8. VERIFY (D1, Max_Info_Frames = V2)
9. WAIT (Internal Processing Fail Time)
10. VERIFY (NP1, Max_Info_Frames = V2)
}

-- verify write via Network Port object

© 2026 by BACnet International. All rights reserved.

11. IF (NP1, Max_Info_Frames is writable) THEN {
12. READ V1 = NP1, Max_Info_Frames
13. VERIFY (D1, Max_Info_Frames = V1)
14. WRITE NP1, Max_Info_Frames = (any valid value V2, other than V1)
15. VERIFY NP1, Changes_Pending = TRUE
16. TRANSMIT ReinitializeDevice-Request
'Reinitialized State of Device' = WARMSTART | ACTIVATE_CHANGES
'Password' = (any valid password)
17. RECEIVE BACnet-SimpleACK-PDU
18. WAIT Activate Changes Fail Time
19. VERIFY NP1, Changes_Pending = FALSE
20. VERIFY (D1, Max_Info_Frames = V2)
}
21. IF (neither D1, Max_Info_Frames nor NP1, Max_Info_Frames is writable, and D1, Max_Info_Frames is not configurable) THEN { -- WriteProperty Service shall not be supported in this case
22. VERIFY (NP1, Max_Info_Frames = 1)
23. VERIFY (D1, Max_Info_Frames = 1)
}

12.3 BACnet/IP Functionality Tests

12.3.2 BBMD B/IP Device with a Server Application

12.3.2.2 Execute Original-Broadcast-NPDU

12.3.2.2.2 Execute Original-Broadcast-NPDU (Two-hop Distribution)

Reason for Change: Possibly a typographical error because the device 'ODIUT' does not exist.

Configuration Requirements: The IUT shall be configured with a BDT that contains:

B/IP Address Broadcast Distribution Mask
IUT 255.255.255.255
BBMD1 255.255.255.255

When the IUT is configured for NAT, the Originating-Device in Forwarded-NPDUs that originate at the IUT, OD, is equal to the Global IP Address and Port of the IUT's Internet Router. When the IUT is not configured for NAT operation, OD is equal to the IUT.

Notes to Tester: The order of the messages transmitted by the IUT is not significant.

Test Steps:

1. TRANSMIT
DA = Local IP Broadcast,
SA = D1,
Original-Broadcast-NPDU,
NPDU = Who-Is
2. RECEIVE
DA = BBMD1,
Forwarded-NPDU,
Originating-Device = D1,
NPDU = Who-Is
3. IF (the IUT responds with Unicast I-Am) THEN

© 2026 by BACnet International. All rights reserved.

RECEIVE DESTINATION = D1,
Original-Unicast-NPDU,
NPDU = I-Am
ELSE
RECEIVE
DA = Local IP Broadcast,
Original-Broadcast-NPDU,
NPDU = I-Am
RECEIVE
DA=BBMD1,
Forwarded-NPDU,
Originating-Device = ~~OD~~IUT,
NPDU = I-Am

12.3.6 Foreign Device Management

12.3.6.3 Foreign Device Table Timer Operations

12.3.6.3.1 Non-Zero-Duration Foreign Device Table Timer Operations

Reason for change: refining the value for Remaining-Time.

Purpose: To verify that the IUT will handle FDT timer operations: finite time Foreign Device registration, re-registration, adding grace period to the supplied Time-To-Live parameter and FDT entry clearing upon timer expiration.

Configuration Requirements: The TD shall take the role of foreign device FD2. The IUT's FDT must be empty. The Network Port object for the BACnet/IP network is NP.

Notes to Tester: The accuracy of the FDT timer shall be specified by the vendor.

Test Steps:

1. TRANSMIT
   DA = IUT,
   SA = FD2,
   Register-Foreign-Device,
   'Time-To-Live' = 60
2. RECEIVE
   DA = FD2,
   SA = IUT,
   BVLC-Result,
   'Result Code' = 0
3. WAIT (10 seconds)
4. TRANSMIT
   DA = IUT,
   SA = FD2,
   Read-Foreign-Device-Table
5. RECEIVE
   DA = FD2,
   SA = IUT,
   Read-Foreign-Device-Table-Ack,
   B/IP address of FD2,
   Time-To-Live = 60,
   ~~Remaining-Time = 80 minus test execution time. (50 is also acceptable if Protocol_Revision < 7)~~

© 2026 by BACnet International. All rights reserved.

Remaining-Time = 90 (Time-To-Live + grace period) - time since FD registration
-- (50 is also acceptable if Protocol_Revision < 7)

6. IF Protocol_Revision >= 17 THEN
   VERIFY NP, BBMD_Foreign_Device_Table = ((B/IP address of FD2, 60, ~~80~~ *90* - ~~execution time~~ *time since FD registration*))
7. TRANSMIT
   DA = IUT,
   SA = FD2,
   Register-Foreign-Device,
   'Time-To-Live' = 40
8. RECEIVE
   DA = FD2,
   SA = IUT,
   BVLC-Result,
   'Result Code' = 0
9. WAIT (30 seconds)
10. TRANSMIT
    DA = IUT,
    SA = FD2,
    Read-Foreign-Device-Table
11. RECEIVE
    DA = FD2,
    SA = IUT,
    Read-Foreign-Device-Table-Ack,
    B/IP address of FD2,
    Time-To-Live = 40,
    Remaining-Time = ~~40 minus test execution time~~ *70 (Time-To-Live + grace period) - time since FD registration*
    -- (10 is also acceptable if Protocol_Revision < 7)
12. IF Protocol_Revision >= 17 THEN
    VERIFY NP, BBMD_Foreign_Device_Table = ((B/IP address of FD2, 40, ~~40~~ *70* - ~~execution time~~ *time since FD registration*))
13. WAIT (50 seconds)
14. TRANSMIT
    DA = IUT,
    SA = FD2,
    Read-Foreign-Device-Table
15. RECEIVE
    DA = FD2,
    SA = IUT,
    Read-Foreign-Device-Table-Ack,
    (No FDT entries)
16. IF Protocol_Revision >= 17 THEN
    VERIFY NP, BBMD_Foreign_Device_Table = ( )

12.3.8 Foreign Device Tests

12.3.8.5 Transmits a Broadcast at Startup preceded by Register-Foreign-Device

Reason for Change: Remove BVLC Result response (step 6) from test per BTL-CR-0610.

Purpose: Verify that mode for use of Register-Foreign-Device and setting of 'BBMD Address' parameter are persistent across reset, and that the issuance of Register-Foreign-Device precedes the first issuance of any broadcast, when in that mode.

© 2026 by BACnet International. All rights reserved.

Test Concept: IUT is put in a mode to use Register-Foreign-Device requests persistently so it will be re-established, then IUT is reset, and the timing of Register-Foreign-Device request to re-establish that precedes the first issuance of any broadcast.

Configuration Requirements: The product's setting of 'BBMD Address' parameter is configured as BBMD1. BBMD1 is the TD simulating a correctly functioning BBMD implementation.

Notes to Tester: For the I-Am, one can precede the Register-Foreign-Device command, as long as then after the Register-Foreign-Device occurs, it is followed by a Distribute-Broadcast-To-Network again, of that I-Am.

Test Steps:

1. MAKE (IUT enter mode for use of Register-Foreign-Device requests, persistently
   so it will be re-established after any reset)
2. MAKE (IUT reset)
3. RECEIVE DA = BBMD1,
   Register-Foreign-Device
4. TRANSMIT BVLC-Result,
   'Result Code' = Successful completion
5. RECEIVE DA = BBMD1,
   Distribute-Broadcast-To-Network,
   NPDU = (any broadcast)
~~6. TRANSMIT BVLC-Result,~~
~~'Result Code' = Successful completion~~

12.3.11 BBMD Configuration Tests – B side

12.3.11.4 *IPv4* - Broadcast Distribution Table Configuration via ~~Hostname~~*Host Name* Entries

Reason for Change: Renamed test, small fixes, and fixed hostname to match standard.

Purpose: Verify that the IUT accepts and resolves *host name*~~hostname~~ entries in the BBMD_Broadcast_Distribution_Table and that the resolved IP *addresses* ~~address~~ are shown in the result of a Read-Broadcast-Distribution-Table request.

Test Concept: Fill the BBMD_Broadcast_Distribution_Table with 4 entries: the IUT, an entry with an IP address (IP1), an entry with a resolvable *host name*~~hostname~~, HN1 that resolves to an IP address, IP2, and an entry with a non-resolvable *host name*~~hostname~~ (HN2). Send a broadcast that the IUT should distribute to its peer BBMDs and verify that it sends to the resolvable entries. Verify that the Broadcast Distribution Table contains the correct entries.

Configuration Requirements: The IUT is configured to operate as a BBMD and *D1*~~the TD (D1)~~ is located on the same IP subnet.

Notes to Tester: The Forwarded-NPDU messages can be received in any order.

Test Steps:

1. WRITE BBMD_Broadcast_Distribution_Table = (4 entries:
   the IUT,
   IP1 (an entry with an IP address),
   HN1 (an entry with a resolvable *host name*~~hostname~~),
   HN2 (an entry with a non-resolvable *host name*~~hostname~~))
2. READ BDT = BBMD_Broadcast_Distribution_Table
3. *VERIFY*~~CHECK~~ (BDT contains IUT, IP1, HN1, HN2 in any order)
4. TRANSMIT ReinitializeDevice-Request
   'Reinitialized State of Device' = ACTIVATE_CHANGES

© 2026 by BACnet International. All rights reserved.

5. WAIT **Activate Changes Fail Time**
6. WAIT until the IUT completes DNS resolution
7. TRANSMIT
   DA = Local IP Broadcast,
   SA = D1,
   Original-Broadcast-NPDU,
   NPDU = Who-Is-Request
8. RECEIVE
   DA = IP1,
   SA = IUT,
   Forwarded-NPDU,
   Originating-Device = D1,
   NPDU = Who-Is
9. RECEIVE
   DA = IP2,
   SA = IUT,
   Forwarded-NPDU,
   Originating-Device = D1,
   NPDU = Who-Is
10. READ BDT = BBMD_Broadcast_Distribution_Table
11. *VERIFY*~~CHECK~~ (BDT contains IUT, IP1, HN1, HN2 in any order)
12. TRANSMIT
    DA = IUT,
    SA = D1,
    Read-Broadcast-Distribution-Table
13. RECEIVE Read-Broadcast-Distribution-Table-Ack,
    'List of BDT Entries' = (4 entries:
    the IUT's IP address,
    the IP address entry,
    the IP address for the resolved *host name*~~hostname~~ entry,
    X'000000000000' for the non-resolvable entry
    -- in the same order as read from the BBMD_Broadcast_Distribution_Table)

12.4 BACnet/IPv6 Functionality Tests

12.4.4 BBMD Tests

This group of tests verifies that a B/IPv6 device that is configured as a BACnet Broadcast Management Device (BBMD) will correctly process incoming B/IPv6 messages that pertain to BBMDs. Only devices that are configured to support BBMD functionality shall execute these tests.

Configuration Requirements: The IUT's Network Port object that represents the B/IPv6 port under test shall be configured as follows:

- BACnet_IPv6_Mode is BBMD
- BACnet_IPv6_Multicast_Address is FF02::BAC0 (Link Local Multicast Address)
- BBMD_Broadcast_Distribution_Table shall contain:

B/IPv6-address
BBMD1
BBMD2
BBMD3

Unless otherwise specified, the TD shall operate as BBMD1.

© 2026 by BACnet International. All rights reserved.

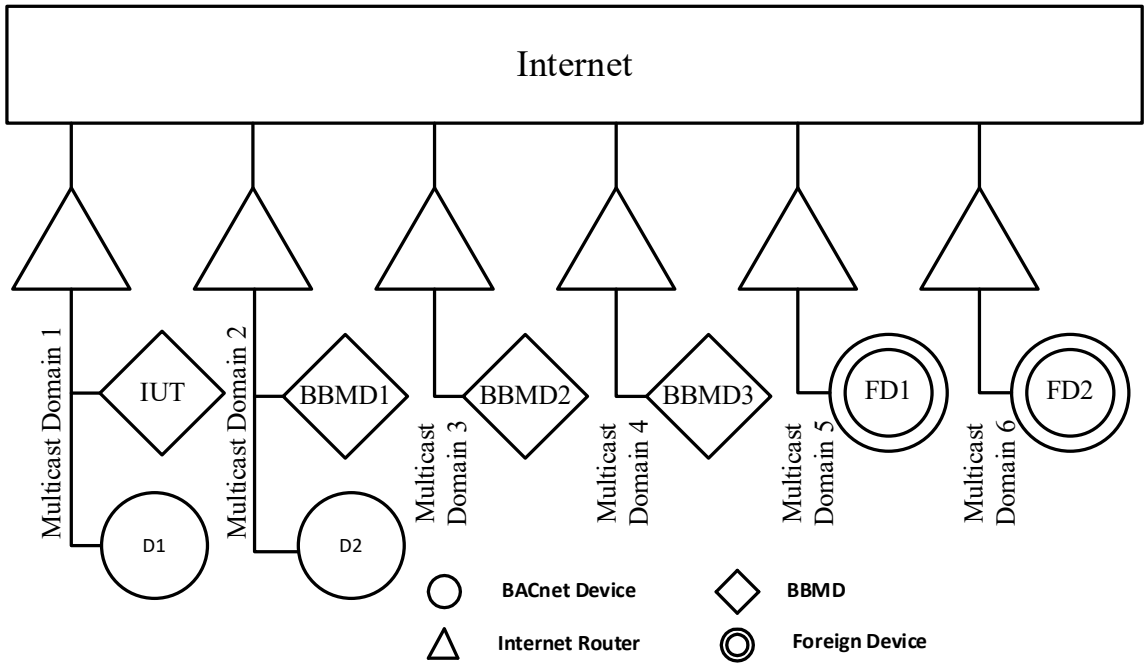


Figure 12-x Logical network configuration for BBMD tests

12.4.4.1 Positive Tests

[Delete text for this section.]

~~This group of tests verifies that a B/IPv6 device that is configured as a BACnet Broadcast Management Device (BBMD) will correctly process incoming B/IPv6 messages that pertain to BBMDs. Only devices that are configured to support BBMD functionality shall execute these tests.~~

~~Configuration Requirements: The IUT's Network Port object that represents the B/IPv6 port under test shall be configured as follows:~~

- ~~BACnet_IPv6_Mode is BBMD~~
- ~~BACnet_IPv6_Multicast_Address is FF02::BAC0 (Link Local Multicast Address)~~
- ~~BBMD_Broadcast_Distribution_Table shall contain:~~

~~**bbmd-address**~~
~~BBMD1~~
~~BBMD2~~
~~BBMD3~~

~~For purposes of these tests, TD shall be operating as BBMD1.~~

12.4.4.1.1 Original-Broadcast-NPDU

Reason For Change: TD is not correctly defined.

Purpose: To verify that the IUT, configured as a BBMD, will forward an Original-Broadcast-NPDU request.

Configuration Requirements: The TD is a non-BBMD and on the same network as the IUT.

Test Steps:

© 2026 by BACnet International. All rights reserved.

1. TRANSMIT
   DA = B/IPv6 Link Local Multicast Address,
   SA = TD,
   Source-Virtual-Address = TD,
   Original-Broadcast-NPDU,
   Who-Is-Request
2. RECEIVE
   DA = BBMD1,
   SA = IUT,
   Forwarded-NPDU,
   Original-Source-Virtual-Address = TD
   Original-Source-B/IPv6-Address = TD
   Who-Is-Request
3. RECEIVE
   DA = BBMD2,
   SA = IUT,
   Forwarded-NPDU,
   Original-Source-Virtual-Address = TD
   Original-Source-B/IPv6-Address = TD
   Who-Is-Request
4. RECEIVE
   DA = BBMD3,
   SA = IUT,
   Forwarded-NPDU,
   Original-Source-Virtual-Address = TD
   Original-Source-B/IPv6-Address = TD
   Who-Is-Request

12.4.4.1.4 Forwarded-Address-Resolution

Reason for Change: Fix the destination address in the last step. Add Test Concept and update Configuration Requirements.

Purpose: To verify that the IUT, configured as a BBMD, will process a Forwarded-Address-Resolution request when the target virtual address is not the virtual address of the IUT.

Test Concept: FD1 shall initiate an Address-Resolution request for FD2. BBMD1 (TD) shall forward the request as a Forwarded-Address-Resolution request to the IUT. The IUT shall forward the request to FD2 and as a local multicast request.

Configuration Requirements: The TD shall operate as BBMD1 and *shall be* listed in the IUTs Broadcast Distribution Table. *FD1 shall be registered as a foreign device with the TD. FD2 shall be registered as a foreign device with the IUT.*

Notes to Tester: For Step 1 to occur, the TD must receive an Address-Resolution request from FD1 requesting the address of FD2. See below.

RECEIVE
DA = TD -- (BBMD1),
SA = FD1,
Address-Resolution,
Source-Virtual-Address = FD1,
Target-Virtual-Address = FD2

© 2026 by BACnet International. All rights reserved.

Notes to Tester: ~~The execution of step 7 is not significant but is shown here in order to demonstrate the completion of the BVLC.~~ The order of the messages *initiated* ~~transmitted~~ by the IUT is not significant. *After step 3, the FD2 would transmit an Address-Resolution-ACK to complete the BVLC transaction, but it is not part of the test. See below.*

TRANSMIT
DA = FD1,
SA = FD2,
Address-Resolution-ACK,
Source-Virtual-Address = FD2,
Destination-Virtual-Address = FD1

Test Steps:

1. TRANSMIT
   DA = IUT,
   SA = TD,
   Forwarded-Address-Resolution,
   Original-Source-Virtual-Address = FD1,
   Target-Virtual-Address = FD2
   Original-Source-B/IPv6-Address = FD1
2. RECEIVE
   DA = B/IPv6 Link Local Multicast Address,
   SA = IUT,
   Forwarded-Address-Resolution,
   Original-Source-Virtual-Address = FD1,
   Target-Virtual-Address = FD2,
   Original-Source-B/IPv6-Address = FD1
3. RECEIVE
   DA = FD2,
   SA = IUT,
   Forwarded-Address-Resolution,
   Original-Source-Virtual-Address = FD1,
   Target-Virtual-Address = FD2,
   Original-Source-B/IPv6-Address = FD1

~~5. TRANSMIT~~
~~DA = TD,~~
~~SA = FD2,~~
~~Address-Resolution-ACK,~~
~~Source-Virtual-Address = FD2,~~
~~Destination-Virtual-Address = TD~~

12.4.4.3 Broadcast Distribution Table Operations

[Remove test 12.4.4.3.1]

12.4.4.3.X1 IPv6 - Broadcast Distribution Table Configuration via Host Name Entries

Reason for Change: No test for this functionality.

Purpose: Verify that the IUT accepts and resolves host name entries in the BBMD_Broadcast_Distribution_Table.

Test Concept: Fill the BBMD_Broadcast_Distribution_Table with 3 entries: an entry with an IPv6 address (IP1), an entry with a resolvable host name, HN1 that resolves to an IPv6 address, IP2, and an entry with a non-resolvable host name (HN2). Send a broadcast that the IUT should distribute to its peer BBMDs and verify that it sends them to the resolvable entries. Verify that the Broadcast Distribution Table contains the correct entries.

© 2026 by BACnet International. All rights reserved.

Configuration Requirements: The IUT is configured to operate as a BBMD and the D1 is located on the same IP subnet.

Notes to Tester: The Forwarded-NPDU messages can be received in any order.

Test Steps:

1. WRITE BBMD_Broadcast_Distribution_Table =(3 entries:
   IP1 (an entry with an IPv6 address),
   HN1 (an entry with a resolvable host name),
   HN2 (an entry with a non-resolvable host name))
2. READ BDT = BBMD_Broadcast_Distribution_Table
3. VERIFY (BDT contains IP1, HN1, HN2 in any order)
4. TRANSMIT ReinitializeDevice-Request
   'Reinitialized State of Device' = ACTIVATE_CHANGES
5. WAIT Activate Changes Fail Time
6. WAIT until the IUT completes DNS resolution
7. TRANSMIT
   DA = Local IP Broadcast,
   SA = D1,
   Original-Broadcast-NPDU,
   NPDU = Who-Is-Request
8. RECEIVE
   DA = IP1,
   SA = IUT,
   Forwarded-NPDU,
   Originating-Device = D1,
   NPDU = Who-Is
9. RECEIVE
   DA = IP2,
   SA = IUT,
   Forwarded-NPDU,
   Originating-Device = D1,
   NPDU = Who-Is
10. READ BDT = BBMD_Broadcast_Distribution_Table
11. VERIFY (BDT contains IP1, HN1, HN2 in any order)

12.4.5 Foreign Device Management Tests

12.4.5.1 Execute Register-Foreign-Device

Reason for change: Step 3 requested the wrong address of a non-existing device.

Purpose: To verify that the IUT will handle a Register-Foreign-Device request.

Test Steps:

1. TRANSMIT
   DA = IUT,
   SA = TD,
   Source-Virtual-Address = TD,
   Register-Foreign-Device,
   'Time-To-Live' = 60
2. RECEIVE

© 2026 by BACnet International. All rights reserved.

DA = TD,
SA = IUT,
Source-Virtual-Address = IUT,
BVLC-Result,
'Result Code' = 0

3. VERIFY NP, BBMD_Foreign_Device_Table = ((B/IPv6 address of ~~FD2~~ *TD*, 60, 90-execution time))

12.5 Secure Connect Functionality Tests

12.5.1 Basic Node Tests

12.5.1.1 Basic Node Positive Tests

12.5.1.1.16 Heartbeat-Request Initiation Test

Reason for Change: modified per Addendum 135-2020cc-1. Changed test to just check the heartbeat interval

Purpose: To verify that the device initiates heartbeats as per its config.

Test Concept: With the IUT connected to the BACnet/SC network, *make the IUT stop initiating requests except heartbeat requests* wait *up to* the IUT's configured heart-beat interval plus 10 seconds and verify ~~that~~ the IUT sent a Heartbeat-Request. *Wait the IUT's configured heart-beat interval plus 10 seconds and verify the IUT sent a Heartbeat-Request at the correct interval.*~~, ensuring that no BVLCs are sent to the IUT during that period.~~ If the IUT claims Protocol_Revision 24 or greater heartbeat interval is the Network Port object, SC_Heartbeat_Timeout property.

Configuration Requirements: ~~Place the IUT in a mode where it will not initiate requests for a period longer than the heartbeat interval (except for the heartbeat request).~~ If the IUT does not support DM-DCC-B and cannot be otherwise configured to behave in this manner, this test shall be skipped.

Test Steps:

1. *MAKE (The IUT stop initiating any requests to the TD except for Heartbeat-Requests)*
2. *BEFORE IUT's heartbeat interval + 10s*
3. *RECEIVE Heartbeat-Request,*
   'Message ID' = (M1: any valid value),
   -- 'Originating Virtual Address' absent
   -- 'Destination Virtual Address' absent
   'Destination Options' = (absent or any valid value),
   -- 'Data Options' absent
4. *TRANSMIT Heartbeat-ACK,*
   'Message ID' = M1,
   -- 'Originating Virtual Address' absent
   -- 'Destination Virtual Address' absent
   'Destination Options' = (absent or any valid value),
   -- 'Data Options' absent
5. *T1 = (the time the Heartbeat-Request was received)*
6. *BEFORE IUT's heartbeat interval + 10s*
7. *RECEIVE Heartbeat-Request,*
   'Message ID' = (M2: any valid value including M1),
   -- 'Originating Virtual Address' absent
   -- 'Destination Virtual Address' absent
   'Destination Options' = (absent or any valid value),
   -- 'Data Options' absent
8. *TRANSMIT Heartbeat-ACK,*

© 2026 by BACnet International. All rights reserved.

'Message ID' = M2,
-- 'Originating Virtual Address' absent
-- 'Destination Virtual Address' absent
'Destination Options' = (absent or any valid value),
-- 'Data Options' absent

9. T2 = (the time the Heartbeat-Request was received)
10. CHECK (T2 - T1 ~= heartbeat interval)

~~1. REPEAT N = (1..Z) DO {~~
~~2. TRANSMIT Encapsulated-NPDU,~~
~~'Message ID' = (M1: any valid value),~~
~~'Originating Virtual Address' = (OVA: any valid value, including absent),~~
~~'Destination Virtual Address' absent~~
~~'Destination Options' (absent or any valid value),~~
~~'Data Options' = ({ X'41'}), Secure Path~~
~~'BACnet NPDU' =~~
~~ReadProperty-Request,~~
~~'Object Identifier' = (the IUT's Device object),~~
~~'Property Identifier' = Object_Name~~
~~3. RECEIVE Encapsulated-NPDU,~~
~~'Message ID' = M1,~~
~~'Originating Virtual Address' absent~~
~~'Destination Virtual Address' = OVA,~~
~~'Destination Options' (absent or any valid value),~~
~~'Data Options' = ({ X'41' or a list of valid header options including Secure Path}),~~
~~'BACnet NPDU' =~~
~~ReadProperty-ACK,~~
~~'Object Identifier' = (the IUT's Device object),~~
~~'Property Identifier' = Object_Name,~~
~~'Property Value' = (the IUT's device object name)~~
~~}~~
~~4. BEFORE ½ of IUT's heartbeat interval + 10s~~
~~5. RECEIVE Heartbeat-Request,~~
~~'Message ID' = (M2: any valid value),~~
~~'Originating Virtual Address' absent~~
~~-- 'Destination Virtual Address' absent~~
~~'Destination Options' = (absent or any valid value),~~
~~'Data Options' absent~~
~~6. TRANSMIT Heartbeat-ACK,~~
~~'Message ID' = M2,~~
~~'Originating Virtual Address' absent~~
~~-- 'Destination Virtual Address' absent~~
~~'Destination Options' = (absent or any valid value),~~
~~'Data Options' absent~~

12.5.1.2 Basic Node Negative Tests

12.5.1.2.10 Node Heartbeat-Request Initialization Failure Test

Reason for Change: Changed the test to just test that the IUT initiates a Disconnect request when the heartbeat request is not acknowledged.

Purpose: To verify that a Node will disconnect if a Heartbeat-ACK is not received.

Test Concept: With the IUT connected to the TD as the primary hub, allow the IUT to connect to the TD. *Make the IUT initiate a Heartbeat-Request and make sure the TD does not respond to the request. Wait the vendor specified timeout for the*

© 2026 by BACnet International. All rights reserved.

IUT to initiate a Disconnect-Request. ~~OD sends a ReadProperty request to the IUT every HB / 2 seconds. HB is the value of the IUTs SC_Heartbeat_Timeout property. Stop sending messages to the IUT. Wait HB plus 10 seconds and verify the IUT sends a Heartbeat-Request, times out waiting for a Heartbeat-ACK and then the IUT sends a Disconnect-Request.~~

Configuration Requirements: ~~Configure the SC_Heartbeat_Timeout property of the TD to be 2 times HB.~~ Place the TD in a mode where it will not respond to Heartbeat-Requests.

Notes to Tester: To make the IUT initiate a Heartbeat-Request, make the TD not initiate or respond to any requests from the IUT.

Test Steps:

~~1. REPEAT N = (1..Z) {~~
~~2. TRANSMIT Encapsulated-NPDU,~~
~~'Message ID' = (M: any valid value),~~
~~'Originating Virtual Address' = (OD's VMAC),~~
~~-- 'Destination Virtual Address' absent~~
~~'Destination Options' (absent or any valid value),~~
~~'Data Options' = ({ X'41'}), Secure Path~~
~~'BACnet NPDU' =~~
~~ReadProperty-Request,~~
~~'Object Identifier' = (the IUT's Device object),~~
~~'Property Identifier' = Object_Name~~
~~3. RECEIVE Encapsulated-NPDU,~~
~~'Message ID' = M,~~
~~'Originating Virtual Address' absent~~
~~'Destination Virtual Address' = (OD's VMAC),~~
~~'Destination Options' (absent or any valid value),~~
~~'Data Options' = ({ X'41' or a list of valid header options including Secure Path}),~~
~~'BACnet NPDU' =~~
~~ReadProperty-ACK,~~
~~'Object Identifier' = (the IUT's Device object),~~
~~'Property Identifier' = Object_Name,~~
~~'Property Value' = (the IUT's device object name)~~
~~4. WAIT HB / 2~~
~~}~~
~~Since we already waited ½ of HB, only HB / 2 of that interval is now given for the IUT to generate a Heartbeat-Request~~
~~5. BEFORE HB / 2 + 10s~~

1. BEFORE heartbeat timeout + 10s
2. RECEIVE Heartbeat-Request,
'Message ID' = (~~M:~~ any valid value)
-- 'Originating Virtual Address' absent
-- 'Destination Virtual Address' absent
'Destination Options' = (absent or any valid value),
-- 'Data Options' absent
~~6~~*3.* BEFORE (2 seconds or Vendor specified timeout)
~~7~~*4.* RECEIVE Disconnect-Request,
'Message ID' = (~~M:~~ any valid value)
-- 'Originating Virtual Address' absent
-- 'Destination Virtual Address' absent
'Destination Options' = (absent or any valid value),
-- 'Data Options' absent

© 2026 by BACnet International. All rights reserved.

12.5.2 Hub Tests

12.5.2.1 Hub Positive Tests

12.5.2.1.9 Duplicate Connection Test

Reason for Change: The test mandates a specific order of connect and disconnect requests when duplicate connection requests occur. See CRR-0528.

Purpose: To verify that duplicate hub connection requests result in the original connection being dropped.

Test Concept: With the IUT operating as hub, connect device D3 to the IUT's hub URI. While maintaining this first connection, make D3 establish a second connection to the IUTs hub URI with the same VMAC. Verify that the IUT accepts the second connect request and closes the first connection. Then, while maintaining the second connection, make D3 establish another new connection (third) with a new VMAC for D3 ensure that the new connect-request is accepted and the second connection is closed.

Notes to Tester: For Steps 6, 7, and 8 and Steps 12, 13, and 14, the order in which the IUT transitions from the existing connection to the new connection is not significant.

Test Steps:

1. MAKE(D3 connect to the IUT's hub function)
2. TRANSMIT PORT (D3-IUT hub first WebSocket),
   Connect-Request,
   'Message ID' = (M1: any valid value),
   -- 'Originating Virtual Address' absent
   -- 'Destination Virtual Address' absent
   -- 'Destination Options' absent
   -- 'Data Options' absent
   'VMAC Address' = (D3's VMAC),
   'Device UUID' = (D3's UUID),
   'Maximum BVLC Length' = (the D3's maximum BVLC accepted length),
   'Maximum NPDU Length' = (the D3's maximum NPDU accepted length)
3. RECEIVE PORT (D3-IUT hub first WebSocket),
   Connect-Accept,
   'Message ID' = M1,
   -- 'Originating Virtual Address' absent
   -- 'Destination Virtual Address' absent
   'VMAC Address' = (IUT's VMAC),
   'Device UUID' = (IUT's UUID),
   'Maximum BVLC Length' = (the IUT's maximum BVLC accepted length),
   'Maximum NPDU Length' = (the IUT's maximum NPDU accepted length)
4. MAKE(~~D1~~*D3* connect a second WebSocket to the IUT's hub function)
5. TRANSMIT PORT (D3-IUT hub second WebSocket),
   Connect-Request,
   'Message ID' = (M2: any valid value),
   -- 'Originating Virtual Address' absent
   -- 'Destination Virtual Address' absent
   -- 'Destination Options' absent
   -- 'Data Options' absent
   'VMAC Address' = (D3's VMAC),
   'Device UUID' = (D3's UUID),
   'Maximum BVLC Length' = (the D3's maximum BVLC accepted length),
   'Maximum NPDU Length' = (the D3's maximum NPDU accepted length)
6. RECEIVE PORT (D3-IUT hub second WebSocket),

© 2026 by BACnet International. All rights reserved.

Connect-Accept,
'Message ID' = M2,
-- 'Originating Virtual Address' absent
-- 'Destination Virtual Address' absent
'Destination Options' = (absent or a list of valid options),
-- 'Data Options' absent
'VMAC Address' = (IUT's VMAC),
'Device UUID' = (IUT's UUID),
'Maximum BVLC Length' = (the IUT's maximum BVLC accepted length),
'Maximum NPDU Length' = (the IUT's maximum NPDU accepted length)

7. RECEIVE PORT (D3-IUT hub first WebSocket),
   Disconnect-Request,
   'Message ID' = M3,
   -- 'Originating Virtual Address' absent
   -- 'Destination Virtual Address' absent
   'Destination Options' = (absent or a list of valid options),
   -- 'Data Options' absent
8. TRANSMIT PORT (D3-IUT hub first WebSocket),
   Disconnect-ACK,
   'Message ID' = M3,
   -- 'Originating Virtual Address' absent
   -- 'Destination Virtual Address' absent
   -- 'Destination Options' absent
   -- 'Data Options' absent
9. CHECK(that the IUT closed D3's initial WebSocket)
10. MAKE(D3 connect a third WebSocket to the IUT's hub function)
11. TRANSMIT PORT (D3-IUT hub second WebSocket),
    Connect-Request,
    'Message ID' = (M4: any valid value),
    -- 'Originating Virtual Address' absent
    -- 'Destination Virtual Address' absent
    -- 'Destination Options' absent
    -- 'Data Options' absent
    'VMAC Address' = (a new VMAC for D3 which does not conflict with any other VMACs),
    'Device UUID' = (D3's UUID),
    'Maximum BVLC Length' = (the D3's maximum BVLC accepted length),
    'Maximum NPDU Length' = (the D3's maximum NPDU accepted length)
12. RECEIVE PORT (D3-IUT hub third WebSocket),
    Connect-Accept,
    'Message ID' = M4,
    -- 'Originating Virtual Address' absent
    -- 'Destination Virtual Address' absent
    'Destination Options' = (absent or a list of valid options),
    -- 'Data Options' absent
    'VMAC Address' = (IUT's VMAC),
    'Device UUID' = (IUT's UUID),
    'Maximum BVLC Length' = (the IUT's maximum BVLC accepted length),
    'Maximum NPDU Length' = (the IUT's maximum NPDU accepted length)
13. RECEIVE PORT (D3-IUT hub second WebSocket),
    Disconnect-Request,
    'Message ID' = M5,
    -- 'Originating Virtual Address' absent
    -- 'Destination Virtual Address' absent
    'Destination Options' = (absent or a list of valid options),
    -- 'Data Options' absent
14. TRANSMIT PORT (D3-IUT hub second WebSocket),

© 2026 by BACnet International. All rights reserved.

Disconnect-ACK,
'Message ID' = M5,
-- 'Originating Virtual Address' absent
-- 'Destination Virtual Address' absent
-- 'Destination Options' absent
-- 'Data Options' absent

15. CHECK(that the IUT closed D3's second WebSocket)

12.5.2.1.11 SC_Hub_Function_Enable Property Test

Reason for change: Fix steps 12 and 13 to correctly come from TD to IUT.

Purpose: To ensure the IUTs hub function can be enabled and disabled using the SC_Hub_Function_Enable property.

Test Concept: With the IUTs SC_Hub_Function_Enable property set to TRUE, verify the IUT is operating as a hub. Change the IUTs SC_Hub_Function_Enable property to FALSE and verify the IUT is no longer operating as a hub.

Configuration Requirements: The IUT is configured as the primary hub and the value of the SC_Hub_Function_Enable property to TRUE. The TD is configured as the failover hub. The TDs primary hub URI is configured to reference the IUT, and the IUTs failover hub URI is configured to reference the TD.

Test Steps:

1. MAKE(the TD open a WebSocket to the IUT's hub function)
2. TRANSMIT PORT (TD-IUT hub WebSocket)
   Connect-Request,
   'Message ID' = (M1: any valid value),
   -- 'Originating Virtual Address' absent
   -- 'Destination Virtual Address' absent
   -- 'Destination Options' absent
   -- 'Data Options' absent
   'VMAC Address' = (TD's VMAC),
   'Device UUID' = (TD's UUID),
   'Maximum BVLC Length' = (the TD's maximum BVLC accepted length),
   'Maximum NPDU Length' = (the TD's maximum NPDU accepted length)
3. RECEIVE PORT (TD-IUT hub WebSocket)
   Connect-Accept,
   'Message ID' = M1,
   -- 'Originating Virtual Address' absent
   -- 'Destination Virtual Address' absent
   'VMAC Address' = (IUT's VMAC),
   'Device UUID' = (IUT's UUID),
   'Maximum BVLC Length' = (the IUT's maximum BVLC accepted length),
   'Maximum NPDU Length' = (the IUT's maximum NPDU accepted length)
4. IF (SC_Hub_Function_Enable is writable) THEN
5. WRITE SC_Hub_Function_Enable = FALSE
6. TRANSMIT ReinitializeDevice-Request
   'Reinitialized State of Device' = WARMSTART | ACTIVATE_CHANGES
   'Password' = (any valid password)
7. RECEIVE BACnet-SimpleACK-PDU
   ELSE
8. MAKE (SC_Hub_Function_Enable = FALSE)
9. WAIT Activate Changes Fail Time
10. CHECK(that the TD attempts and fails to open a WebSocket to the IUT)

© 2026 by BACnet International. All rights reserved.

11. CHECK(that the IUT opens a WebSocket with the TD)
12. *RECEIVE*~~TRANSMIT~~ PORT (~~TD-IUT~~*IUT-TD failover* hub WebSocket)
    Connect-Request,
    'Message ID' = (M1: any valid value),
    -- 'Originating Virtual Address' absent
    -- 'Destination Virtual Address' absent
    -- 'Destination Options' absent
    -- 'Data Options' absent
    'VMAC Address' = ~~(TD's VMAC)~~*(IUT's VMAC)*,
    'Device UUID' = ~~(TD's UUID)~~*(IUT's UUID)*,
    'Maximum BVLC Length' = (the ~~TD's~~*IUT's* maximum BVLC accepted length),
    'Maximum NPDU Length' = ~~(~~the ~~TD's~~*IUT's* maximum NPDU accepted length)
13. ~~RECEIVE~~*TRANSMIT* PORT (~~TD-IUT~~*IUT-TD failover* hub WebSocket)
    Connect-Accept,
    'Message ID' = ~~M1~~*M2*,
    -- 'Originating Virtual Address' absent
    -- 'Destination Virtual Address' absent
    'VMAC Address' = ~~(~~IUT's*TD's* VMAC),
    'Device UUID' = (~~IUT's~~*TD's* UUID),
    'Maximum BVLC Length' = (the ~~IUT's~~*TD's* maximum BVLC accepted length),
    'Maximum NPDU Length' = (the ~~IUT's~~*TD's* maximum NPDU accepted length)
14. VERIFY (SC_Hub_Function_Enable = FALSE)
15. IF (SC_Hub_Function_Enable is writable) THEN
16. WRITE SC_Hub_Function_Enable = TRUE
17. TRANSMIT ReinitializeDevice-Request
        'Reinitialized State of Device' = WARMSTART | ACTIVATE_CHANGES
        'Password' = (any valid password)
18. RECEIVE BACnet-SimpleACK-PDU
    ELSE
19. MAKE (SC_Hub_Function_Enable = TRUE)
20. WAIT Activate Changes Fail Time
21. MAKE(the TD open a WebSocket to the IUT's hub function)
22. TRANSMIT PORT (TD-IUT hub WebSocket)
    Connect-Request,
    'Message ID' = (M1: any valid value),
    -- 'Originating Virtual Address' absent
    -- 'Destination Virtual Address' absent
    -- 'Destination Options' absent
    -- 'Data Options' absent
    'VMAC Address' = (TD's VMAC),
    'Device UUID' = (TD's UUID),
    'Maximum BVLC Length' = (the TD's maximum BVLC accepted length),
    'Maximum NPDU Length' = (the TD's maximum NPDU accepted length)
23. RECEIVE PORT (TD-IUT hub WebSocket)
    Connect-Accept,
    'Message ID' = M1,
    -- 'Originating Virtual Address' absent
    -- 'Destination Virtual Address' absent
    'VMAC Address' = (IUT's VMAC),
    'Device UUID' = (IUT's UUID),
    'Maximum BVLC Length' = (the IUT's maximum BVLC accepted length),
    'Maximum NPDU Length' = (the IUT's maximum NPDU accepted length)

© 2026 by BACnet International. All rights reserved.

12.5.2.2 Hub Negative Tests

12.5.2.2.5 Malformed BVLC Test for Nodes operating as a Hub

Reason for Change: Added Unknown BVLC function check with DVA. Fixed addressing in Step 13(10) and 14(11).

Purpose: Verify the device NAKs malformed / unknown unicast BVLCs and ignores malformed / unknown broadcast BVLCs when operating as a Node with an active hub function.

Test Concept: With the IUT connected to the BACnet/SC network and operating as the active hub, send a sequence of malformed unicast and broadcast BVLCs to the IUT. Verify that the IUT responds as required to each message and does not process nor route the messages.

Configuration Requirements: The IUT is connected to the BACnet/SC network and is operating as the active hub. The TD is connected to the BACnet/SC network as a node.

Test Steps:

-- Unknown BVLC function -- *no DVA*
1. TRANSMIT
   'BVLC Function' = (IV: an unknown 1-octet value),
   'Message ID' = (M1: any valid value),
   -- 'Originating Virtual Address' absent
   -- 'Destination Virtual Address' absent
   -- 'Destination Options' absent
   -- 'Data Options' absent
2. RECEIVE BVLC-Result,
   'Message ID' = M1,
   -- 'Originating Virtual Address' absent
   -- 'Destination Virtual Address' absent
   'Destination Options' = (absent or a valid list of options),
   -- 'Data Options' absent
   'Result for BVLC Function' = IV,-- the supplied unknown BVLC function from the request
   'Result Code' = X'01', -- NAK
   'Error Header Marker' = X'00', -- not a header option problem
   'Error Class' = COMMUNICATION,
   'Error Code' = BVLC_FUNCTION_UNKNOWN
3. CHECK(that the IUT did not process nor forward the request)

-- Unknown BVLC function – valid DVA
4. TRANSMIT
   'BVLC Function' = (IV: an unknown 1-octet value),
   'Message ID' = (M1: any valid value),
   -- 'Originating Virtual Address' absent
   'Destination Virtual Address' = (IUT's VMAC),
   -- 'Destination Options' absent
   -- 'Data Options' absent
5. RECEIVE BVLC-Result,
   'Message ID' = M1,
   'Originating Virtual Address' = (IUT's VMAC),
   -- 'Destination Virtual Address' absent
   'Destination Options' = (absent or a valid list of options),
   -- 'Data Options' absent
   'Result for BVLC Function' = IV,-- the supplied unknown BVLC function from the request
   'Result Code' = X'01', -- NAK
   'Error Header Marker' = X'00', -- not a header option problem

© 2026 by BACnet International. All rights reserved.

'Error Class' = COMMUNICATION,
'Error Code' = BVLC_FUNCTION_UNKNOWN
6. CHECK(that the IUT did not process nor forward the request)

-- Inclusion of an Originating Virtual Address when it is required to be absent
~~4~~*7*. TRANSMIT Disconnect-Request,
'Message ID' = (M2: any valid value),
'Originating Virtual Address' = D3, -- error condition, required to be absent
-- 'Destination Virtual Address' absent
'Destination Options' = (absent or a valid list of options),
-- 'Data Options' absent
~~5~~*8*. IF (receive a message) THEN
~~6~~*9*. RECEIVE BVLC-Result,
'Message ID' = M2,
-- 'Originating Virtual Address' absent
-- A valid message from a hub requires DVA to be absent
-- 'Destination Virtual Address' absent
'Destination Options' = (absent or a valid list of options),
-- 'Data Options' absent
'Result for BVLC Function' = X'08', -- Disconnect-Request
'Result Code' = X'01', -- NAK
'Error Header Marker' = X'00', -- not a header option problem
'Error Class' = (COMMUNICATION),
'Error Code' = (HEADER_ENCODING_ERROR,
INCONSISTENT_PARAMETERS,
PARAMETER_OUT_OF_RANGE or OTHER)
~~7~~*10*.CHECK(that the IUT did not process the request)

-- Inclusion of a 'Destination Virtual Address when it is required to be absent
~~8~~*11*.TRANSMIT Disconnect-Request,
'Message ID' = (M3: any valid value),
-- 'Originating Virtual Address' absent,
'Destination Virtual Address' = (IUT's VMAC), -- error condition, required to be absent
'Destination Options' = (absent or a valid list of options),
-- 'Data Options' absent
~~9~~*12*.IF (receive a message) THEN
~~10~~*13*. RECEIVE BVLC-Result,
'Message ID' = M3,
-- 'Originating Virtual Address' absent
-- 'Destination Virtual Address' absent
'Destination Options' = (absent or a valid list of options),
-- 'Data Options' absent
'Result for BVLC Function' = X'08', -- Disconnect-Request
'Result Code' = X'01', -- NAK
'Error Header Marker' = X'00', -- not a header option problem
'Error Class' = (COMMUNICATION),
'Error Code' = (HEADER_ENCODING_ERROR,
INCONSISTENT_PARAMETERS,
PARAMETER_OUT_OF_RANGE or OTHER)
~~11~~*14*. CHECK(that the IUT did not process the request)

-- A truncated message
~~12~~*15*. TRANSMIT Encapsulated-NPDU,
'Message ID' = (M4: any valid value),
--'Originating Virtual Address' absent
'Destination Virtual Address' = (IUT's VMAC),

© 2026 by BACnet International. All rights reserved.

~~'Destination Virtual Address' absent,~~
-- 'Destination Options' absent
-- 'Data Options' absent
-- no NPDU included in the message
~~13~~*16*. RECEIVE BVLC-Result,
'Message ID' = M4,
'Originating Virtual Address' = (IUT's VMAC),
~~'Originating Virtual Address' absent~~
-- 'Destination Virtual Address' absent
'Destination Options' = (absent or a valid list of options),
-- 'Data Options' absent
'Result for BVLC Function' = X'01', -- Encapsulated-NPDU
'Result Code' = X'01', -- NAK
'Error Header Marker' = X'00', -- not a header option problem
'Error Class' = COMMUNICATION,
'Error Code' = MESSAGE_INCOMPLETE | PAYLOAD_EXPECTED
~~14~~*17*. CHECK(that the IUT did not process the request)

-- A message with extra octets added on
~~15~~*18*. TRANSMIT Disconnect-Request,
'Message ID' = (M5: any valid value),
-- 'Originating Virtual Address' absent
-- 'Destination Virtual Address' absent
-- 'Destination Options' absent
-- 'Data Options' absent
(extra octets) = ({ X'C1', --a bunch of octets that look like valid data options.
X'BF0003000012',
X'3F0003000034'})
~~16~~*19*. RECEIVE BVLC-Result,
'Message ID' = M5,
-- 'Originating Virtual Address' absent
-- 'Destination Virtual Address' absent
'Destination Options' = (absent or a valid list of options),
-- 'Data Options' absent
'Result for BVLC Function' = X'08', -- Disconnect-Request
'Result Code' = X'01', -- NAK
'Error Header Marker' = X'00', -- not a header option problem
'Error Class' = COMMUNICATION,
'Error Code' = UNEXPECTED_DATA
~~17~~*20*. CHECK(that the IUT did not process the request)

-- A truncated broadcast message
~~18~~*21*. TRANSMIT Encapsulated-NPDU,
DESTINATION = GLOBAL_BROADCAST,
'Message ID' = (M6: any valid value),
-- 'Originating Virtual Address' absent,
'Destination Virtual Address' = (local broadcast VMAC),
-- 'Destination Options' absent
'Data Options' = ({ X'E1' -- Secure Path,
more follows (but none are present)
})
~~19~~*22*. CHECK(that the IUT does not send a BVLC-Result, did not process the message, nor route the message)

© 2026 by BACnet International. All rights reserved.

12.5.2.2.6 Hub Heartbeat-Request Initialization Failure Test

Reason for Change: Changed the test to just test that the IUT initiates a Disconnect request when the heartbeat request is not acknowledged.
Purpose: To verify that a Hub initiates a Heartbeat-Request before attempting to terminate a connection.

Test Concept: With the IUT operating as a hub, the TD connects to the IUT. *Make the IUT initiate a Heartbeat-Request and make sure the TD does not respond to the request. Wait the vendor specified timeout for the IUT to initiate a Disconnect-Request. If the IUT's hub function does not have a mechanism to determine if the connection or the connection peer might not be alive, this test shall be skipped.* ~~The TD sends a ReadProperty request to the IUT every HB / 2 seconds. HB is the value of the IUTs SC_Heartbeat_Timeout property. Stop sending messages to the IUT. Wait HB plus 10 seconds and verify the IUT sends a Heartbeat-Request, times out waiting for a Heartbeat-ACK and then the IUT sends a Disconnect-Request.~~

Configuration Requirements: ~~Configure the SC_Heartbeat_Timeout property of the TD to be 2 times HB.~~ Place the TD in a mode where it will not respond to Heartbeat-Requests.

Test Steps:

~~1. REPEAT N = (1..Z) {~~
~~2. TRANSMIT Encapsulated-NPDU,~~
~~'Message ID' = (M: any valid value),~~
~~'Originating Virtual Address' absent~~
~~'Destination Virtual Address' = (IUT's VMAC),~~
~~'Destination Options' (absent or any valid value),~~
~~'Data Options' = ({ X'41'}), Secure Path~~
~~'BACnet NPDU' =~~
~~ReadProperty-Request,~~
~~'Object Identifier' = (the IUT's Device object),~~
~~'Property Identifier' = Object_Name~~
~~3. RECEIVE Encapsulated-NPDU,~~
~~'Message ID' = M,~~
~~'Originating Virtual Address' = (IUT's VMAC),~~
~~'Destination Virtual Address' absent~~
~~'Destination Options' (absent or any valid value),~~
~~'Data Options' = ({ X'41' or a list of valid header options including Secure Path}),~~
~~'BACnet NPDU' =~~
~~ReadProperty-ACK,~~
~~'Object Identifier' = (the IUT's Device object),~~
~~'Property Identifier' = Object_Name,~~
~~'Property Value' = (the IUT's device object name)~~
~~4. WAIT HB / 2~~
~~}~~
~~Since we already waited ½ of HB, only HB / 2 of that interval is now given for the IUT to generate a Heartbeat-Request~~
~~5. BEFORE HB / 2 + 10s~~
1. MAKE (the IUT initiate a Heartbeat-Request)
~~6~~*2*. RECEIVE Heartbeat-Request,
'Message ID' = (~~M:~~ any valid value)
-- 'Originating Virtual Address' absent
-- 'Destination Virtual Address' absent
'Destination Options' = (absent or any valid value),
-- 'Data Options' absent
~~7~~*3*. BEFORE (2 seconds or Vendor specified timeout)
~~8~~*4*. RECEIVE Disconnect-Request,
'Message ID' = (~~M:~~ any valid value)
-- 'Originating Virtual Address' absent

© 2026 by BACnet International. All rights reserved.

-- 'Destination Virtual Address' absent
'Destination Options' = (absent or any valid value),
-- 'Data Options' absent

12.5.3 Direct Connect Tests

12.5.3.3 Initiating Direct Connect Tests

12.5.3.3.2 Initiating Direct Connect Negative Tests

12.5.3.3.2.3 Rejection of Invalid Certificate Outgoing Connection Test

Reason for Change: Remove requirement to run the test if the accepting device does not have a valid certificate.

Purpose: To verify that the IUT will drop initiated connection attempts if the peer's certificate *has expired* ~~is invalid~~.

Test Concept: With the IUT configured to initiate direct connections. Make the IUT attempt to connect to D3 via a direct connection. D3 presents an *expired* ~~invalid~~ certificate during the *direct connect* connection *attempt*. Verify that the WebSocket is not established.

Configuration Requirements: The IUT is configured to initiate direct connections *to D3*. D3 is configured with *an expiring certificate* ~~an invalid certificate~~. D3 shall be configured to accept the IUT's certificate.

Test Steps:

1. IF (IUT requires D3 to be connected to the hub) THEN
2. MAKE(D3 establish a connection to the hub)
3. WAIT (for D3's certificate to expire)
~~1~~*4*. MAKE(the IUT attempt to establish a direct connection to D3)
~~2~~*5*. CHECK(that the IUT initiated a WebSocket connection)
~~3~~*6*. CHECK(that the WebSocket connection was failed by the IUT)

13 SPECIAL FUNCTIONALITY TESTS

13.2 Time Manager

[Remove test 13.2.1]

13.5 Subordinate Proxy Tests

13.5.1 Manual Subordinate Binding Test

Reason for Change: Added Device_Address_Proxy_Table (PR 27 requirements)

Purpose: This test verifies that the IUT can find and confirm MS/TP subordinate devices listed in the Manual_Subordinate_Binding property in the IUT's device object. This test also verifies that the IUT correctly distinguishes between subordinate and manager devices, and that it performs periodic confirmation of subordinate devices.

Test Concept: Configure the Manual_Subordinate_Binding property with the address of two MS/TP devices. Attach a subordinate at one of the addresses and a manager that supports the Who-Is and I-Am services at the other address. Monitor the network to verify that the IUT confirms the devices and then verify that the subordinate device address is added into the Subordinate_Address_Binding *and Device_Address_Proxy_Table properties* ~~property~~ and that the manager address is not. The subordinate is then removed, and once the IUT re-confirms the subordinate, it is verified that the subordinate is removed from the Subordinate_Address_Binding *and Device_Address_Proxy_Table properties* ~~property~~.

© 2026 by BACnet International. All rights reserved.

Configuration Requirements: The MS/TP network shall contain a subordinate device at address A1 with a device identifier of D1 and a manager device at address A2 with a device identifier of D2. The subordinate device shall not support the reading of its device object using the wildcard instance of 4194303. The manager device shall execute the Who-Is service and initiate the I-Am service. The IUT shall be configured to perform subordinate proxying.

Notes to Tester: The ReadProperty requests can be initiated in any order.

Test Steps:

1. WRITE Subordinate_Proxy_Enable = TRUE
~~1~~*2.* BEFORE **Subordinate Proxy Confirm Interval**
3. RECEIVE
DESTINATION=A1,
SRC=IUT,
ReadProperty-Request,
'Object Identifier' = (D1),
'Property Identifier' = Protocol_Services_Supported
4. TRANSMIT
DESTINATION=IUT,
SRC=A1,
ReadProperty-ACK
'Object-Identifier' = (D1),
'Property Identifier' = Protocol_Services_Supported,
'Property Value' = (any valid value for this property)
5. RECEIVE
DESTINATION=A2,
SRC=IUT,
ReadProperty-Request,
'Object Identifier' = (D2),
'Property Identifier' = Protocol_Services_Supported
6. TRANSMIT
DESTINATION=IUT,
SRC=A2,
ReadProperty-ACK
'Object-Identifier' = (D2),
'Property Identifier' = Protocol_Services_Supported,
'Property Value' = (any valid value for this property)
~~2. REPEAT addr=(A1, A2) DO {~~
~~3. RECEIVE DESTINATION=addr, SRC=IUT~~
~~ReadProperty-Request,~~
~~'Object Identifier' = (the correct value for the address being queried),~~
~~'Property Identifier' = Protocol_Services_Supported~~
~~4. RECEIVE DESTINATION=IUT, SRC=addr~~
~~BACnet-Complex-ACK,~~
~~'Object Identifier' = (the correct value for the address being queried),~~
~~'Property Identifier' = Protocol_Services_Supported,~~
~~'Property Value' = (any valid value for this property)~~
~~}~~
~~5~~*7.* VERIFY Subordinate_Address_Binding = ((A1, D1))
8. IF Protocol_Revision >= 27 THEN
9. VERIFY Device_Address_Proxy_Table = (contains (A1, D1))
-- Remove the subordinate device from the MS/TP network
~~6~~*10.*BEFORE **Subordinate Proxy Confirm Interval**
11. RECEIVE
DESTINATION=A1,

© 2026 by BACnet International. All rights reserved.

SRC=IUT,
ReadProperty-Request,
'Object Identifier' = (D1 | 4194303),
'Property Identifier' = Protocol_Services_Supported
~~7. RECEIVE DESTINATION=A1, SRC=IUT~~
~~ReadProperty-Request,~~
~~'Object Identifier' = (DEVICE,the correct value for the address being queried),~~
~~'Property Identifier' = Protocol_Services_Supported~~
-- note that the subordinate will not reply to this request as it is no longer connected to the network.
~~8~~*12.* WAIT (longer than it takes for the IUT to timeout on this request)
~~9~~*13.* VERIFY Subordinate_Address_Binding = ()
14. IF Protocol_Revision >= 27 THEN
15. VERIFY Device_Address_Proxy_Table = (does not contain (A1, D1))

Passing Result: The IUT shall read the Protocol_Services_Supported property from each MS/TP device to determine whether or not it supports Who-Is. The implementer may have chosen to read this property after the IUT has determined that a device exists at a given MAC address. In this case, any property shall be accepted in *the ReadProperty request* ~~step 1~~, and the TD shall expect a subsequent read from the IUT for the Protocol_Services_Supported property for the two devices attached to the MS/TP segment. The IUT is also allowed to generate any other traffic it cares to during the test including, but not limited to, reading property values from the devices it finds.

13.5.2 Automatic Subordinate Discovery Test

Reason for Change: Added Device_Address_Proxy_Table (PR 27 requirements)

Purpose: This test verifies that an IUT that contains an Auto_Subordinate_Discovery property is able to find and confirm MS/TP subordinates that support reading with the wildcard instance of 4194303. This test also verifies that the IUT correctly distinguishes between subordinate and manager devices, and that it performs periodic confirmation of subordinate devices.

Test Concept: Configure the Auto_Subordinate_Binding property to enable automatic detection of subordinates. Connect a subordinate and a manager to the MS/TP network. Monitor the network to verify that the IUT searches for and finds each device connected to the MS/TP segment. Verify that the IUT added the subordinate to its Subordinate_Address_Binding *and Device_Address_Proxy_Table properties* ~~property~~ and that it did not add the manager to the list. A subordinate is then removed, and once the IUT re-confirms the subordinate, it is verified that the subordinate is removed from the Subordinate_Address_Binding *and Device_Address_Proxy_Table properties* ~~property~~.

Configuration Requirements: The MS/TP network shall contain a subordinate device at address A1 with a device identifier of D1 and a manager device at address A2 with a device identifier of D2. The subordinate device shall support the reading of its device object using the wildcard instance of 4194303. The manager device shall execute the Who-Is service and initiate the I-Am service. The IUT shall be configured to perform automatic subordinate detection on all MS/TP addresses except its own and the broadcast address.

Test Steps:

1. WRITE Subordinate_Proxy_Enable = TRUE
~~1. MAKE(the IUT start its automatic subordinate detection)~~
~~2. BEFORE **Subordinate Proxy Confirm Interval**~~
*2. BEFORE **Subordinate Proxy Confirm Interval** {*
3. REPEAT addr=(all MS/TP addresses excluding the IUT's MAC address) DO {
4. RECEIVE
DESTINATION=addr,
SRC=IUT,
ReadProperty-Request,
'Object Identifier' = (Device, 4194303),
'Property Identifier' = Protocol_Services_Supported
5. IF (addr is present on the network) THEN {
6. TRANSMIT

© 2026 by BACnet International. All rights reserved.

DESTINATION=IUT,
SRC=addr,
ReadProperty-ACK
'Object-Identifier' = (any valid value),
'Property Identifier' = Protocol_Services_Supported,
'Property Value' = (any valid value for this property)
}
ELSE
7. WAIT (for the IUT to move to the next addr. This may include Number_Of_APDU_Retries attempts)
}
}
~~RECEIVE DESTINATION=addr, SRC=IUT~~
~~ReadProperty-Request,~~
~~'Object Identifier' = (DEVICE,4194303),~~
~~'Property Identifier' = Protocol_Services_Supported~~
~~RECEIVE DESTINATION=IUT, SRC=addr~~
~~BACnet-Complex-ACK,~~
~~'Object Identifier' = (the correct value for the address being queried),~~
~~'Property Identifier' = Protocol_Services_Supported,~~
~~'Property Value' = (any valid value for this property)~~
~~}~~
~~3~~*8.* VERIFY Subordinate_Address_Binding = ((A1, D1))
9. IF Protocol_Revision >= 27 THEN
10. VERIFY Device_Address_Proxy_Table = (contains (A1, D1))
11. MAKE (remove the subordinate device from the MS/TP network)
12. BEFORE ***Subordinate Proxy Confirm Interval***
13. RECEIVE
DESTINATION=A1,
SRC=IUT,
ReadProperty-Request,
'Object Identifier' = (D1 | 4194303),
'Property Identifier' = Protocol_Services_Supported
~~4. Remove the subordinate device from the MS/TP network~~
~~5. BEFORE **Subordinate Proxy Confirm Interval**~~
~~RECEIVE DESTINATION=A1, SRC=IUT~~
~~ReadProperty-Request,~~
~~'Object Identifier' = (DEVICE, the correct value for the address being queried),~~
~~'Property Identifier' = Protocol_Services_Supported~~
-- note that the subordinate will not reply to this request as it is no longer connected to the network.
~~6~~*14.*WAIT (longer than it takes the IUT to timeout on this request)
~~7~~*15.*VERIFY Subordinate_Address_Binding = ()
16. IF Protocol_Revision >= 27 THEN
17. VERIFY Device_Address_Proxy_Table = (does not contain (A1, D1))

Passing Result: The IUT shall read the Protocol_Services_Supported property from each MS/TP device to determine whether or not it supports Who-Is. The implementer may have chosen to read this property after the IUT has determined that a device exists in a given slot. In this case, any property shall be accepted in *the ReadProperty request* ~~step 1~~, and the TD shall expect a subsequent read from the IUT for the Protocol_Services_Supported property for the two devices attached to the MS/TP segment. The IUT is also allowed to generate any other traffic it cares to during the test including, but not limited to, reading property values from the devices it finds.

13.10 Workstation Scheduling Tests

Purpose: This group of tests verifies that the IUT is capable of viewing and modifying existing ~~schedules,~~ *Schedule, Calendar and Timer objects*.

© 2026 by BACnet International. All rights reserved.

Test Concept: This test consists of high-level MAKE and CHECK steps that are expected to be manually executed while monitoring BACnet communications using a BACnet network analyzer.

Configuration Requirements: The reference device shall be configured to indicate that it supports only the ReadProperty-Request and WriteProperty-Request services in the Protocol_Services_Supported property of its Device object. (Service clients that can use services more complex than ReadProperty-Request, such as ReadPropertyMultiple-Request, shall be able to adapt to a server device that does not support these more complex services.) The reference device is configured to contain Schedules*, Calendars, and Timers* ~~and Calendars~~ as specified by S1, S2, C1, *C2, and T1* ~~and C2~~. The datatype of the 'value' portion of BACnetTimeValue can be varied to test scheduling of different datatypes, but all of the BACnetTimeValue elements shall be consistent within the same Schedule object, and the Present_Value and properties referenced by List_Of_Object_Property_References shall also be of the same datatype. The reference objects S1, S2, C1, *C2, and T1* ~~and C2~~ represent the standard test data, but the tester is free to use additional test data for this test.

~~Note: The reference Schedule and Calendars contain some data that requires support for Protocol_Revision 4 or later. To convert the Schedule to conform to Protocol_Revision 3 or older, make the following changes:~~

1. ~~Change month 13 to month 3 in BACnetSpecialEvent[5].~~
2. ~~Remove the Schedule_Default property.~~
3. ~~Change month 14 to month 2 in BACnetSpecialEvent[6].~~
4. ~~Change dayOfWeek from 32 to 28 in BACnetSpecialEvent[6].~~
5. ~~Change the last two CalendarEntries in the Date_List of Calendar C1 to remove the use of special values indicating all even months, all odd months, and the Last Day of the month.~~

[Add to Clause 13.10, between Reference Schedule S3 and Reference Calendar C1]

Reference Schedule S4:

Identical to schedule S1, except that the Exception_Schedule property is empty.

[Add to the end of Clause 13.10]
Reference Timer T1:

State_Change_Values = {
<value1>,
Null, -- indicates Relinquish
<value2>,
<value3>,
[1] Null, -- indicates no-value
<value4>,
<value5>}

13.10.4 Modify an Exception_Schedule

13.10.4.7 Modify an Exception_Schedule by Adding a BACnetSpecialEvent with Period of Choice calendarEntry of choice Date

Reason for Change: Resolve issues with adding a schedule to a full reference schedule. Using a new empty reference schedule S4.

Purpose: Demonstrate that the IUT can accept user input and modify the Exception_Schedule by adding a BACnetSpecialEvent of the schedule with the calendarEntry of type Date.

© 2026 by BACnet International. All rights reserved.

Configuration Requirements: A reference device contains reference objects ~~S1~~*S4*, C1, and C2.

Test Steps:

1. MAKE (the IUT add an entire BACnetSpecialEvent with period of choice calendarEntry of choice Date to the Exception_Schedule of ~~S1~~*S4*)
2. CHECK (Did the IUT write the modified Exception_Schedule correctly?)

Notes to Tester: For example, try different Date representations that leave one or more of each of the following fields unspecified: year, month, dayOfMonth, dayOfWeek. If the IUT supports devices with Protocol_Revision >= 4, try using the special values for month 13 (all odd) and 14 (all even) and the special value of 32 (last day) for the dayOfMonth.

13.10.4.8 Modify an Exception_Schedule by Adding a BACnetSpecialEvent with Period of Choice calendarEntry of Choice DateRange

Reason for Change: Resolve issues with adding a schedule to a full reference schedule. Using a new empty reference schedule S4.

Purpose: Demonstrate that the IUT can accept user input and modify the Exception_Schedule by adding a BACnetSpecialEvent of the schedule with the calendarEntry of type DateRange.

Configuration Requirements: A reference device contains reference objects ~~S1~~*S4*, C1, and C2.

Test Steps:

1. MAKE (the IUT add an entire BACnetSpecialEvent with period of choice calendarEntry of choice DateRange to the Exception_Schedule of ~~S1~~*S4*)
2. CHECK (Did the IUT write the modified Exception_Schedule correctly?)

Notes to Tester: It is not required that the IUT be capable of creating DateRanges with wildcard fields with the exception of unspecified dates (all fields equal to 255 in either one or both of the date fields of the BACnetDateRange).

13.10.4.9 Modify an Exception_Schedule by Adding a BACnetSpecialEvent with Period of Choice calendarEntry of Choice WeekNDay

Reason for Change: Resolve issues with adding a schedule to a full reference schedule. Using a new empty reference schedule S4.

Purpose: Demonstrate that the IUT can accept user input and modify the Exception_Schedule by adding a BACnetSpecialEvent of the schedule with the calendarEntry of type WeekNDay.

Test Concept: For different WeekNDay values, add a BACnetCalendarEntry to the Date_List property of an Exception_Schedule, and verify the IUT writes the modified Date_List correctly to the reference device.

Configuration Requirements: A reference device contains reference objects ~~S1~~*S4*, C1, and C2. Choose the value set to use from the table below based on the protocol-revision for the IUT.

		WeekNDay = <Month><weekOfMonth><dayOfWeek>		
Value Set	Protocol_Revision	Month	weekOfMonth	dayOfWeek
1	PR < 4	0 to 12 and FF	1 to 6 and FF	1 to 7 and FF
2	PR >= 4	0 to 12 and FF and 13 and 14	1 to 6 and FF	1 to 7 and FF
3	PR >= 18	0 to 12 and FF and 13 and 14	1 to 9 and FF	1 to 7 and FF

Test Steps:

1. REPEAT WND = (WeekNDay Value Set from table) DO {

© 2026 by BACnet International. All rights reserved.

2. MAKE (the IUT add an entire BACnetSpecialEvent with period of choice calendarEntry of choice WeekNDay with value of WND to the Exception_Schedule of ~~S1~~*S4*)
3. CHECK (Did the IUT write the modified Exception_Schedule correctly?)

}

13.10.4.10 Modify an Exception_Schedule by Adding a BACnetSpecialEvent with Period of Choice calendarReference

Reason for Change: Resolve issues with adding a schedule to a full reference schedule. Using a new empty reference schedule S4.

Purpose: Demonstrate that the IUT can accept user input and modify the Exception_Schedule by adding a BACnetSpecialEvent of the schedule with the calendarEntry of type calendarReference.

Configuration Requirements: A reference device contains reference objects ~~S1~~*S4*, C1, and C2.

Test Steps:

1. MAKE (the IUT add an entire BACnetSpecialEvent with period of choice calendarReference to the Exception_Schedule of ~~S1~~*S4*)
2. CHECK (Did the IUT write the modified Exception_Schedule correctly?)

Notes to Tester: For example, add a BACnetSpecialEvent that is another CalendarReference to C2.

13.10.X Modify a Timer Object

This clause is used to verify that the IUT can present and modify a Timer object in a server device.

13.10.X.1 Read and Present a State_Change_Values Property

Purpose: Demonstrate that the IUT reads and presents the State_Change_Values property of a Timer object.

Configuration Requirements: A reference device contains Timer object T1.

Test Steps:

1. MAKE (the IUT read and present the data represented by the State_Change_Values of T1)
2. CHECK (Did the IUT properly present the data represented by the State_Change_Values?)

13.10.X.2 Modify a State_Change_Values Property

Purpose: Demonstrate that the IUT can accept user input and use it to modify the State_Change_Values and List_Of_Object_Property_References properties.

Configuration Requirements: A reference device contains a Timer object (T1) with writable State_Change_Values and List_Of_Object_Property_References properties.

Test Steps:

1. MAKE (the IUT write the State_Change_Values property with a set of values of data type DT1. Ensure at least one value is Null and one is no-value)
2. MAKE (the IUT write List_Of_Object_Property_References with a reference that accepts data type DT1)
3. CHECK (Did the IUT write the change to the State_Change_Values and List_Of_Object_Property_References properties correctly?)

© 2026 by BACnet International. All rights reserved.

13.X1 Time Series Data Tests

13.X1.1 Time Series Data Export Tests

13.X1.1.1 Validate Time Series Data Export Test

Reason for Change: No test for this functionality.

Purpose: This test verifies the IUT can correctly export a Time Series Data csv file where the data is not from a trend log object.

Test Concept: Configure the IUT to contain a Time Series Data csv file with the specified primitive data type in at least one column, C1, of the file. Export the file and check the file is valid.

Configuration Requirements: None.

Test Steps:

1. MAKE (export time series data as a csv file)
2. CHECK (the top row, first column contains "DateTime")
3. CHECK (the top row, remaining columns contain printable strings and are unique within the row)
4. CHECK (the first column after row 1, contains only valid timestamps)
5. CHECK (column C1 contains only the specified primitive data type or missing data as specified in AA.1.2)

13.X2 Device Proxying Functionality Tests

This clause defines the tests required to verify device proxying functionality. The tests assume that the device under test (IUT) has two ports: Port 1, which is directly connected to Network 1, and Port 2, which is directly connected to Network 2. Port 1 represents the BACnet network for which the IUT performs device proxying. Devices that support device proxying on more than one datalink shall be tested using these procedures, with Port 1 configured for each applicable datalink.

The test descriptions in this clause assume that the Test Device (TD) can connect to both Network 1 and Network 2 simultaneously and can emulate all other devices in the test environment. The TD's connection to Network 1 is referred to as Port A, and its connection to Network 2 as Port B. As an acceptable alternative, an internetwork may be constructed using real devices instead of TD emulation. Devices D1, D2, D3, D4, and D5 must be capable of initiating I-Am requests (i.e., they cannot be subordinate devices).

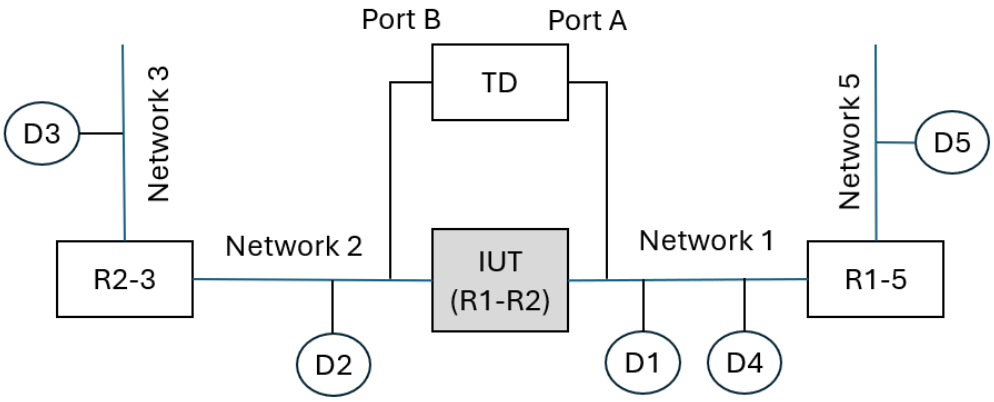


Figure 13-X. Logical internetwork configuration for device proxying functionality tests.

13.X2.1 Device Proxying Test

Reason for Change: No tests exist for this functionality

© 2026 by BACnet International. All rights reserved.

Purpose: To verify the IUT correctly executes Who-Is requests when device proxying is enabled.

Test Concept:

The IUT is configured to proxy for device D1 on Network 1. A broadcast Who-Is request, with a range including D1, is sent to the IUT from D2 and another from D3 to verify the IUT responds with the appropriate unicast I-Am response on behalf of D1 and does not initiate any I-Am requests on Network 1.
A broadcast Who-Is request, with a range that does not include D1, is sent to the IUT from D2 to verify the IUT does not respond on behalf of D1.

Configuration Requirements: Device_Address_Proxy_Enable = True in the network port object, NP, corresponding to the proxied network (Network 1).

Test Steps:

1. VERIFY NP, Device_Address_Proxy_Enable = TRUE
2. TRANSMIT
   DA = LOCAL BROADCAST
   SA = D1
   I-Am Request,
   'Device Identifier' = (D1's Device Identifier),
   'Max APDU Length Accepted' = (any valid value),
   'Segmentation Supported' = (any valid value),
   'Vendor Identifier' = (TD's Vendor_Identifier
3. WAIT (10 seconds)
4. VERIFY Device_Address_Proxy_Table = (a list of values that contains D1)
5. READ L1 = Device_Address_Proxy_Table[D1]('max-apdu-length-accepted'
6. READ S1 = Device_Address_Proxy_Table[D1]('segmentation-supported")
7. READ V1 = Device_Address_Proxy_Table[D1]('vendor-id')
8. READ N1 = NP, Network_Number
9. TRANSMIT,
   SA = D2
   DA = IUT
   DNET = Network 1
   DADR = BROADCAST
   Who-Is-Request,
   'Device Instance Range Low Limit'= (The Device object instance number of D1),
   'Device Instance Range High Limit' = (The Device object instance number of D1)
10. RECEIVE PORT B,
    DA = D2
    SNET = Network 1
    SADR = (MAC address of D1)
    I-Am Request,
    'Device Identifier' = (D1),
    'Max APDU Length Accepted' = L1
    'Segmentation Supported' = S1,
    'Vendor Identifier' = V1
11. CHECK (the IUT does not transmit the Who-Is-Request on Port A)
12. CHECK (the IUT does not transmit the I-Am Request on Port A)

-- (Who Is from a remote network)
13. TRANSMIT,
    DA = IUT
    SNET = Network 3
    SADR = D3

© 2026 by BACnet International. All rights reserved.

DNET = Network 1
DADR = BROADCAST
Who-Is-Request,
'Device Instance Range Low Limit'= (The Device object instance number of D1),
'Device Instance Range High Limit' = (The Device object instance number of D1)

14. RECEIVE PORT B,
DA = R2-3
SNET = Network 1
SADR = (MAC address of D1)
DNET = Network 3
DADR = D3
I-Am Request,
'Device Identifier' = (D1),
'Max APDU Length Accepted' = L1
'Segmentation Supported' = S1,
'Vendor Identifier' = V1
15. CHECK (the IUT does not transmit the Who-Is-Request on Port A)
16. CHECK (the IUT does not transmit the I-Am Request on Port A)

-- (Who-Is with range that does not include D1)
17. TRANSMIT PORT B,
DA = IUT
DNET = Network 1
DADR = BROADCAST
Who-Is-Request,
'Device Instance Range Low Limit'= (LL, any value <> D1),
'Device Instance Range High Limit' = (LL)
18. CHECK (the IUT does not transmit an I-Am for D1 on Port A or Port B)

13.X2.2 Device Added When Detected via I-Am

Reason for Change: No tests exist for this functionality

Purpose: To verify that the IUT will add a new device to its Device_Address_Proxy_Table, when it receives an I-Am that originated from the proxied network (Network 1).

Test Concept:

The IUT is configured to proxy devices on network 1. An I-Am from device, D1, is transmitted as a local broadcast, and it's verified that D1 is added to the IUT's Device_Address_Proxy_Table. An I-Am from device, D4, is transmitted as a global broadcast. It's verified that D4 is added to the IUT's Device_Address_Proxy_Table.

Configuration Requirements: Device_Address_Proxy_Enable = True in the network port object, NP, corresponding to the proxied network (Network 1). At the start of the test, D1, and D4 are not in the IUT's Device_Address_Proxy_Table. NP is the proxied network's Network Port object.

Test Steps:

1. VERIFY NP, Device_Address_Proxy_Enable = TRUE
2. TRANSMIT,
SA = D1
DA = LOCAL BROADCAST
I-Am Request,
'Device Identifier' = (D1),
'Max APDU Length Accepted' = (any valid value),
'Segmentation Supported' = (any valid value),

© 2026 by BACnet International. All rights reserved.

'Vendor Identifier' = (any valid value)
3. WAIT **Internal Processing Fail Time** + 10s
4. VERIFY Device_Address_Proxy_Table = (a list of values that contains D1)
5. TRANSMIT,
   SA = D4
   DA = LOCAL BROADCAST
   DNET = GLOBAL BROADCAST
   DLEN = 0
   I-Am Request,
   'Device Identifier' = (D4),
   'Max APDU Length Accepted' = (any valid value),
   'Segmentation Supported' = (any valid value),
   'Vendor Identifier' = (any valid value)
6. WAIT **Internal Processing Fail Time** + 10s
7. VERIFY Device_Address_Proxy_Table = (a list of values that contains both D1 and D4)

13.X2.3 Device Discovery Without I-Am

Reason for Change: No tests exist for this functionality

Purpose: To verify that the IUT can detect a device that is added to the network when that device does not initiate unsolicited I-Am's.

Test Concept:

The TD is silently added to the proxied network such that it does not initiate any unsolicited messages. The IUT is made to discover the TD and the Device_Address_Proxy_Table is checked to verify the TD has been added.

Configuration Requirements: Device_Address_Proxy_Enable = True in the network port object, NP, corresponding to the proxied network (Network 1). The TD is configured such that it will not initiate any unsolicited requests. At the start of the test, the IUT is not aware of the TD and there is no entry for it in the Device_Address_Proxy_Table.

Notes to tester: The vendor may provide instructions regarding device discovery that could facilitate execution of this test in a timely manner (for example: if the IUT discovers devices at midnight, adjust the clock to 23:59:00.00).

Test Steps:

1. VERIFY NP, Device_Address_Proxy_Enable = TRUE
2. WAIT (a vendor defined time to discover devices)
3. VERIFY Device_Address_Proxy_Table = (a list of values that contains the TD)

13.X2.4 Update of Last I-Am Time

Reason for Change: No tests exist for this functionality

Purpose: To verify that the Last I-Am time in the Device_Address_Proxy_Table updates when the corresponding device initiates an I-Am.

Test Concept:

The TD initiates an I-Am message on the proxied network of the IUT. The Last I-Am Time portion of the TD's entry in the Device_Address_Proxy_Table is compared with the IUT's local date and time.

Configuration Requirements: Device_Address_Proxy_Enable = True in the network port object, NP, corresponding to the proxied network (Network 1).

© 2026 by BACnet International. All rights reserved.

Test Steps:

1. VERIFY NP, Device_Address_Proxy_Enable = TRUE
2. TRANSMIT PORT A,
   DA = LOCAL BROADCAST
   I-Am Request,
   'Device Identifier' = (TD),
   'Max APDU Length Accepted' = (any valid value),
   'Segmentation Supported' = (any valid value),
   'Vendor Identifier' = (any valid value)
3. READ currentDate = (the IUT's Device object), Local_Date
4. READ currentTime = (the IUT's Device object), Local_Time
5. WAIT **Internal Processing Fail Time**
6. READ T1 = Device_Address_Proxy_Table, 'Last I-Am Time' corresponding to TD's entry
7. VERIFY T1 ~= {currentDate, currentTime}

13.X2.5 Revalidation of Proxied Devices After Restart

Reason for Change: No tests exist for this functionality

Purpose: To verify that the IUT revalidates all devices in the Device_Address_Proxy_Table after a restart and removes offline devices.

Test Concept:

With the IUT proxying for devices D1 and D4, the IUT is reset and D4 is removed from the network. When the IUT has returned to an operational state, it's verified that it attempts to validate the presence of D1 and D4 by initiating unicast Who-Is requests. When D4 fails to respond after 10 seconds, it's verified that the IUT sends another Who-Is for D4. 10 seconds after the second Who-Is for D4, the Device_Address_Proxy_Table is read to verify that it only contains D1.

Configuration Requirements: Device_Address_Proxy_Enable = True in the Network Port object, NP, for the proxied network.

Notes to tester: The sequence of steps 11,12,13 may also be 12,11,13 or 11,13,12

Test Steps:

1. VERIFY NP, Device_Address_Proxy_Enable = TRUE
2. TRANSMIT
   DA = LOCAL BROADCAST
   SA = D1
   I-Am Request,
   'Device Identifier' = (D1's Device Identifier),
   'Max APDU Length Accepted' = (any valid value),
   'Segmentation Supported' = (any valid value),
   'Vendor Identifier' = (TD's Vendor_Identifier
3. TRANSMIT
   DA = LOCAL BROADCAST
   SA = D4
   I-Am Request,
   'Device Identifier' = (D4's Device Identifier),
   'Max APDU Length Accepted' = (any valid value),
   'Segmentation Supported' = (any valid value),
   'Vendor Identifier' = (TD's Vendor_Identifier
4. WAIT (10 seconds)
5. VERIFY Device_Address_Proxy_Table = (a list of values that contains both D1 and D4)
6. IF (IUT supports the ReinitializeDevice service) THEN {

© 2026 by BACnet International. All rights reserved.

7. TRANSMIT ReinitializeDevice-Request
   'Reinitialized State of Device' = WARMSTART
   'Password' = (any valid password) --if required by IUT
   }
8. ELSE
9. MAKE (the IUT restart)
10. Remove D4 from the network
11. WHILE (System_Status <> Operational) {}
12. RECEIVE,
    DA = D1
    Who-Is-Request,
    'Device Instance Range Low Limit'= (any value <= the device object instance number of D1),
    'Device Instance Range High Limit' = (any value >= the device object instance number of D1)
13. RECEIVE,
    DA = D4
    Who-Is-Request,
    'Device Instance Range Low Limit'= (any value <= the device object instance number of D4),
    'Device Instance Range High Limit' = (any value <= the device object instance number of D4)
14. TRANSMIT,
    DA = IUT | LOCAL BROADCAST
    I-Am Request,
    'Device Identifier' = (D1),
    'Max APDU Length Accepted' = (any valid value),
    'Segmentation Supported' = (any valid value),
    'Vendor Identifier' = (any valid value)
15. WAIT **Internal Processing Fail Time** + 10s
16. RECEIVE,
    DA = D4
    Who-Is-Request,
    'Device Instance Range Low Limit'= (any value <= the device object instance number of D4),
    'Device Instance Range High Limit' = (any value <= the device object instance number of D4)
17. WAIT **Internal Processing Fail Time** + 10s
18. VERIFY Device_Address_Proxy_Table = (a list of values that contains D1 and excludes D4)

13.X2.6 Automatic Removal of Offline Devices from Device_Address_Proxy_Table

Reason for Change: No tests exist for this functionality

Purpose: To verify that offline devices are removed from the Device_Address_Proxy_Table on or before Device_Address_Proxy_Timeout is reached.

Test Concept:

A device, D1, is added to the IUT's Device_Address_Proxy_Table and then removed from the network. Before Device_Address_Proxy_Timeout is reached, the IUT detects that D1 is no longer on the network and removes D1 from the Device_Address_Proxy_Table.

Configuration Requirements: Device_Address_Proxy_Enable = True in the Network Port object, NP, for the proxied network.

Notes to Tester: The IUT is permitted to use non-BACnet means to determine the presence or absence of D1 on the network, including monitoring non-BACnet traffic such as ARP-requests, therefore, D1 must be completely removed from the network in Step 5. For this reason, the TD cannot be used for D1.

Test Steps:

1. VERIFY NP, Device_Address_Proxy_Enable = TRUE

© 2026 by BACnet International. All rights reserved.

2. TRANSMIT,
   DA = IUT | LOCAL BROADCAST
   SA = D1
   I-Am Request,
   'Device Identifier' = (D1),
   'Max APDU Length Accepted' = (any valid value),
   'Segmentation Supported' = (any valid value),
   'Vendor Identifier' = (any valid value)
3. WAIT **Internal Processing Fail Time** + 10s
4. VERIFY Device_Address_Proxy_Table = (a list of values that contains D1)
5. MAKE (D1 removed from the network)
6. WAIT (NP, Device_Address_Proxy_Timeout)
7. VERIFY Device_Address_Proxy_Table = (a list of values that does not include D1)
8. TRANSMIT PORT B,
   DA = LOCAL BROADCAST,
   Who-Is Request
9. RECEIVE PORT A,
   SA = IUT
   DA = LOCAL BROADCAST,
   SNET = NETWORK 2
   SADR = (TD's MAC address)
   Who-Is Request
10. CHECK (the IUT does not send I-Am's for D1)

13.X2.7 Processing Who-Is Messages from Proxied Network

Reason for Change: No tests exist for this functionality

Purpose: To verify that the IUT correctly processes Who-Is Messages originating from devices on the proxied network.

Test Concept:

Remote and Global broadcast Who-Is messages, with different ranges, are originated from devices located on the proxied network of the IUT. In the first case, the Who-Is requests have no range, and it's verified that the IUT correctly forwards the Who-Is requests. In the second case, the Who-Is requests have a range but one that does not include any devices in the Device_Address_Proxy_Table, and it's verified that the IUT correctly forwards the Who-Is requests. In the third case, the Who-Is requests have a range matching a device in the Device_Address_Proxy_Table, and it's verified the IUT does not forward the Who-Is requests, nor does it send a I-Am on behalf of the device in it's Device_Address_Proxy_Table.

Configuration Requirements: Device_Address_Proxy_Enable = True in the Network Port object, NP, for the proxied network.

Notes to Tester: When a specific DNET (i.e. DNET = Network 3) is used in the Who-Is request, the IUT may attempt to discover the router to the network before forwarding the Who-Is. If this is the case, the tester must use a real router to the remote network or simulate the response(s) to satisfy the IUT's router discovery mechanism.

Test Steps:

1. VERIFY NP, Device_Address_Proxy_Enable = TRUE
2. TRANSMIT PORT A,
   DA = IUT | LOCAL BROADCAST
   I-Am Request,
   'Device Identifier' = (TD),
   'Max APDU Length Accepted' = (any valid value),
   'Segmentation Supported' = (any valid value),
   'Vendor Identifier' = (any valid value)
3. VERIFY Device_Address_Proxy_Table = (a list of values that contains the TD)

© 2026 by BACnet International. All rights reserved.

-- (case 1a: no range – local broadcast message, Who-Is destined for remote network)
4. TRANSMIT PORT A,
   SA = TD
   DA = LOCAL BROADCAST
   DNET = Network 3
   DLEN = 0
   Who-Is Request
5. RECEIVE PORT B,
   DA = LOCAL BROADCAST | Unicast to router for N2-3
   SNET = Network 1
   SADR = TD
   DNET = Network 3
   DLEN = 0
   Who-Is Request
6. CHECK (the IUT does not forward the Who-Is to Network 1)
7. CHECK (the IUT does not transmit any I-Ams for proxied devices)

-- (case 1b: no range – global broadcast Who-Is)
8. TRANSMIT PORT A,
   SA = TD
   DA = LOCAL BROADCAST
   DNET = GLOBAL BROADCAST
   DLEN = 0
   Who-Is Request
9. RECEIVE PORT B,
   DA = LOCAL BROADCAST
   SADR = TD
   SNET = Network 1
   DNET = GLOBAL BROADCAST
   DLEN = 0
   Who-Is Request
10. CHECK (the IUT does not forward the Who-Is to Network 1)
11. CHECK (the IUT does not send I-Am's for proxied devices)

-- (case 1c: no range – local broadcast)
12. TRANSMIT PORT A,
   SA = TD
   DA = LOCAL BROADCAST
   Who-Is Request
13. CHECK (the IUT does not forward the Who-Is request on Port B)
14. CHECK (the IUT does not send I-Am's for proxied devices)

-- (case 2a: range does not include any devices in the IUT's Device_Address_Proxy_Table, local broadcast message, destined for remote network)
15. TRANSMIT PORT A,
   SA = TD
   DA = LOCAL BROADCAST
   DNET = Network 3
   DLEN = 0
   Who-Is Request,
   'Device Instance Range Low Limit'= (LL, any value not in the IUT's Device_Address_Proxy_Table),
   'Device Instance Range High Limit' = LL
16. RECEIVE PORT B,
   DA = LOCAL BROADCAST | Unicast to router for N2-3
   SNET = Network 1

© 2026 by BACnet International. All rights reserved.

SADR = TD
DNET = Network 3
DLEN = 0
Who-Is Request,
'Device Instance Range Low Limit'= (LL, any value not in the IUT's Device_Address_Proxy_Table),
'Device Instance Range High Limit' = LL

17. CHECK (the IUT does not forward the Who-Is to Network 1)
18. CHECK (the IUT does not transmit any I-Ams for proxied devices)

-- (case 2b: range does not include any devices in the IUT's Device_Address_Proxy_Table, global broadcast)

19. TRANSMIT PORT A,
SA = TD
DA = LOCAL BROADCAST
DNET = GLOBAL BROADCAST
DLEN = 0
Who-Is Request,
'Device Instance Range Low Limit'= (LL, any value not in the IUT's Device_Address_Proxy_Table),
'Device Instance Range High Limit' = LL
20. RECEIVE PORT B,
DA = LOCAL BROADCAST
SADR = TD
SNET = Network 1
DNET = GLOBAL BROADCAST
DLEN = 0
Who-Is Request,
'Device Instance Range Low Limit'= (LL, any value not in the IUT's Device_Address_Proxy_Table),
'Device Instance Range High Limit' = LL
21. CHECK (the IUT does not forward the Who-Is to Network 1)
22. CHECK (the IUT does not transmit any I-Ams for proxied devices)

-- (case 2c: range does not include any devices in the IUT's Device_Address_Proxy_Table, local broadcast)

23. TRANSMIT PORT A,
SA = TD
DA = LOCAL BROADCAST
Who-Is Request,
'Device Instance Range Low Limit'= (LL, any value not in the IUT's Device_Address_Proxy_Table),
'Device Instance Range High Limit' = LL
24. CHECK (the IUT does not forward the Who-Is request on Port B)
25. CHECK (the IUT does not transmit any I-Ams for proxied devices)

-- (case 3a: range specifies D1, global broadcast Who-Is)

26. TRANSMIT PORT A,
SA = TD
DA = LOCAL BROADCAST
DNET = GLOBAL BROADCAST
DLEN = 0
Who-Is Request,
'Device Instance Range Low Limit'= D1
'Device Instance Range High Limit' = D1
27. CHECK (the IUT does not forward the Who-Is to N1)
28. CHECK (the IUT does not send any I-Ams for proxied devices)

-- (case 3b: range specifies D1, local broadcast Who-Is)

29. TRANSMIT PORT A,
SA = TD
DA = LOCAL BROADCAST

© 2026 by BACnet International. All rights reserved.

Who-Is Request,
'Device Instance Range Low Limit'= D1
'Device Instance Range High Limit' = D1
30. CHECK (the IUT does not forward the Who-Is to Network 1)
31. CHECK (the IUT does not send any I-Ams for proxied devices)

13.X2.8 Sending Who-Is Messages Through the Proxied Network

Reason for Change: No tests exist for this functionality

Purpose: To verify that the IUT will correctly pass along a Who-Is request to a router on its proxied network.

Test Concept:

The IUT is made to discover BACnet routers located on it's proxied network. After router discovery is complete, The TD transmits a global broadcast Who-Is on Port B with no range restriction. The IUT sends I-Am's on behalf of it's proxied devices and sends the Who-Is request to the router(s) located on its proxied network. The TD then transmits a global broadcast Who-Is on Port B with range parameters for devices not contained in the Device_Address_Proxy_Table. The IUT transmits the Who-Is request to the router(s) located on its proxied network.

Configuration Requirements: Device_Address_Proxy_Enable = True in the Network Port object, NP, for the proxied network.

Test Steps:

1. VERIFY NP, Device_Address_Proxy_Enable = TRUE
2. MAKE (the IUT discover routers)
3. WAIT (vendor specified time for router discovery to complete)

-- (Global broadcast with no range)
4. TRANSMIT PORT B,
DA = LOCAL BROADCAST
DNET = GLOBAL BROADCAST
DLEN = 0
Who-Is Request
5. IF (NP is not the IUT's home network) {
6. RECEIVE PORT B,
DA = TD | LOCAL BROADCAST | GLOBAL BROADCAST
SA = IUT
I-Am Request,
'Device Identifier' = (the IUT's device object),
'Max APDU Length Accepted' = (any valid value),
'Segmentation Supported' = (any valid value),
'Vendor Identifier' = (any valid value)
}
7. REPEAT X = (all devices in the IUT's Device_Address_Proxy_Table) DO {
RECEIVE PORT B,
DA = TD | LOCAL BROADCAST | GLOBAL BROADCAST
SA = IUT
SADR = (MAC address of X)
SNET = Network 1
I-Am Request,
'Device Identifier' = (X),
'Max APDU Length Accepted' = (any valid value),
'Segmentation Supported' = (any valid value),
'Vendor Identifier' = (any valid value)
}

© 2026 by BACnet International. All rights reserved.

8. RECEIVE PORT A,
   SA = IUT
   DA = R1-5
   DNET = GLOBAL BROADCAST
   DLEN = 0
   Who-Is Request

-- (Global broadcast with range not in proxy table)
9. TRANSMIT PORT B,
   DA = LOCAL BROADCAST
   DNET = GLOBAL BROADCAST
   DLEN = 0
   Who-Is Request,
   'Device Instance Range Low Limit'= (LL, any value not in the IUT's Device_Address_Proxy_Table),
   'Device Instance Range High Limit' = LL

10. RECEIVE PORT A,
    SA = IUT
    DA = R1-5
    DNET = GLOBAL BROADCAST
    DLEN = 0
    Who-Is Request,
    'Device Instance Range Low Limit'= LL,
    'Device Instance Range High Limit' = LL
11. CHECK (the IUT does not send I-Ams for any entry in it's Device_Address_Proxy_Table)

13.X2.9 Entry for Self in Device_Address_Proxy_Table

Reason for Change: No tests exist for this functionality

Purpose: To verify that the IUT's Device Address Proxy Table has an entry for itself when the proxied network is the home network.

Test Concept:

With the IUT configured to perform Device Proxy functionality on its home network, the Device_Address_Proxy_Table is checked to verify that it contains an entry for the IUT and that the parameters for that entry are correct.

Configuration Requirements: Device_Address_Proxy_Enable = True in the Network Port object, NP, for the proxied network. The network represented by NP is considered the home network.

Test Steps:

1. VERIFY NP, Device_Address_Proxy_Enable = TRUE
2. READ N1 = NP, Network_Number
3. READ M1 = NP, MAC_Address
4. READ currentDate = (the IUT's Device object), Local_Date
5. READ currentTime = (the IUT's Device object), Local_Time
6. READ Segmentation = (the IUT's Device object), Segmentation_Supported
7. READ VID = (the IUT's Device object), Vendor_Identifier
8. VERIFY U1 = (the IUT's Device object), Device_Address_Proxy_Table [entry for U1],
   Address = (N1, M1),
   I Am = (the IUT's Device object, any valid value, Segmentation, VID),
   Last I Am Time ~= {currentDate, currentTime}

© 2026 by BACnet International. All rights reserved.

13.X2.10 Proxy Table Cleared when Proxy Enable Set to False

Reason for Change: No tests exist for this functionality

Purpose: To verify that the IUT's Device_Address_Proxy_Table is cleared and device proxy functionality is disabled when the Device_Address_Proxy_Enable property is set to false.

Test Concept:

The IUT's Device_Address_Proxy_Table is populated with multiple entries (D1 and the D4). The Device_Address_Proxy_Enable property is then set to False and the table is read again to verify that it is empty. A Who-Is request, with a range including D1 is sent to the IUT and it's verified the request reaches D1.

Configuration Requirements: Device_Address_Proxy_Enable = True in the Network Port object, NP, for the proxied network.

Test Steps:

1. VERIFY NP, Device_Address_Proxy_Enable = TRUE
2. TRANSMIT,
   DA = IUT | LOCAL BROADCAST
   I-Am Request,
   'Device Identifier' = (D1),
   'Max APDU Length Accepted' = (any valid value),
   'Segmentation Supported' = (any valid value),
   'Vendor Identifier' = (any valid value)
3. TRANSMIT,
   DA = IUT | LOCAL BROADCAST
   I-Am Request,
   'Device Identifier' = (D4),
   'Max APDU Length Accepted' = (any valid value),
   'Segmentation Supported' = (any valid value),
   'Vendor Identifier' = (any valid value)
4. WAIT **Internal Processing Fail Time** + 10s
5. VERIFY Device_Address_Proxy_Table = (a list of values that contains both D1 and D4)
6. WRITE NP, Device_Address_Proxy_Enable = FALSE
7. VERIFY Device_Address_Proxy_Table = (an empty list)
8. TRANSMIT,
   SA = D2
   DA = LOCAL BROADCAST
   DNET = Network 1
   DLEN = 0
   Who-Is Request,
   'Device Instance Range Low Limit'= D1,
   'Device Instance Range High Limit' = D1
9. RECEIVE PORT A,
   SA = IUT
   DA = BROADCAST
   SADR = (MAC address of D2)
   SNET = Network 2
   Who-Is Request,
   'Device Instance Range Low Limit'= D1,
   'Device Instance Range High Limit' = D1
10. TRANSMIT,
    SA = D1
    DA = IUT
    DNET = Network 2

© 2026 by BACnet International. All rights reserved.

DADR = (MAC address of D2)
I-Am Request,
'Device Identifier' = (D1),
'Max APDU Length Accepted' = (any valid value),
'Segmentation Supported' = (any valid value),
'Vendor Identifier' = (any valid value)

11. RECEIVE PORT B,
SA = IUT
DA = D2
SADR = (MAC address of D1)
SNET = Network 1
I-Am Request,
'Device Identifier' = (D1),
'Max APDU Length Accepted' = (any valid value),
'Segmentation Supported' = (any valid value),
'Vendor Identifier' = (any valid value)

13.X2.11 Proxying Multiple Devices Test

Reason for Change: No tests exist for this functionality

Purpose: To verify the IUT can proxy for multiple devices on the proxied network. Standard 135 stipulates 128 proxied devices.

Test Concept: The IUT is proxying at least devices D1 and D4 on Network 1. D2 initiates a Who-Is for all devices on Network 1. Check the IUT initiates I-Am requests for D1 and D4 and all other proxied devices.

Configuration Requirements: Device_Address_Proxy_Enable = True in the Network Port object, NP, for the proxied network. No router exists on Network 1.

Notes to Tester: The I-Ams can be received in any order.

Test Steps:

1. VERIFY NP, Device_Address_Proxy_Enable = TRUE
2. TRANSMIT PORT B,
SA = D2
DA = IUT
DNET = Network 1
DADR = BROADCAST
Who-Is-Request,
3. REPEAT X = (the number of proxied devices on Network 1) {
4. RECEIVE PORT B,
DA = D2
SNET = Network 1
SADR = (MAC address of X)
I-Am Request,
'Device Identifier' = (X),
'Max APDU Length Accepted' = (value specified for X)
'Segmentation Supported' = (value specified for X),
'Vendor Identifier' = (value specified for X)
}

© 2026 by BACnet International. All rights reserved.

13.X2.12 I-Am from Non-Proxied Network Not Added to Table

Reason for Change: No tests exist for this functionality

Purpose: To verify that the IUT will not add a new device to its Device_Address_Proxy_Table when it receives an I-Am from a non-proxied network.

Test Concept:

The IUT is configured to proxy devices on network 1. An I-Am for device, D2 is received by the IUT and it's verified that D2 is not added to the IUT's Device_Address_Proxy_Table. Another I-Am, sent from a remote device, D5, is received by the IUT from a router on the proxied network and it's verified that D5 is not added to the IUT's Device_Address_Proxy_Table.

Configuration Requirements: Device_Address_Proxy_Enable = True in the Network Port object, NP, for the proxied network. At the start of the test, D2 and D5 are not in the IUT's Device_Address_Proxy_Table.

Test Steps:

1. VERIFY NP, Device_Address_Proxy_Enable = TRUE
2. TRANSMIT,
   DA = LOCAL BROADCAST
   I-Am Request,
   'Device Identifier' = (D2),
   'Max APDU Length Accepted' = (any valid value),
   'Segmentation Supported' = (any valid value),
   'Vendor Identifier' = (any valid value)
3. WAIT **Internal Processing Fail Time** + 10s
4. VERIFY Device_Address_Proxy_Table = (a list of values that does not include D2)
5. TRANSMIT,
   SA = R1-5
   DA = LOCAL BROADCAST
   SNET = Network 5
   SADR = (MAC Address of D5)
   DNET = GLOBAL BROADCAST
   DADR = BROADCAST
   I-Am Request,
   'Device Identifier' = (D5),
   'Max APDU Length Accepted' = (any valid value),
   'Segmentation Supported' = (any valid value),
   'Vendor Identifier' = (any valid value)
6. BEFORE **Internal Processing Fail Time**
7. RECEIVE PORT B,
   DA = LOCAL BROADCAST
   SNET = Network 5
   SADR = (MAC address of D5)
   DNET = GLOBAL BROADCAST
   DLEN = 0
   I-Am Request,
   'Device Identifier' = (D5),
   'Max APDU Length Accepted' = (any valid value),
   'Segmentation Supported' = (any valid value),
   'Vendor Identifier' = (any valid value)
8. WAIT (the remainder of **Internal Processing Fail Time**) + 10s
9. VERIFY Device_Address_Proxy_Table = (a list of values that does not include D5)

© 2026 by BACnet International. All rights reserved.

13.X2.13 Device I-Am Forwarding Test

Reason for Change: No tests exist for this functionality

Purpose: To verify the IUT does not forward I-Ams initiated by proxied devices.

Test Concept: The IUT is proxying the TD on Network 1. The TD initiates a global broadcast I-Am and it is verified the IUT does not initiate an I-Am for the TD on Network 2 because of the initial I-Am.

Configuration Requirements: Device_Address_Proxy_Enable = True in the Network Port object, NP, for the proxied network

Test Steps:

1. VERIFY NP, Device_Address_Proxy_Enable = TRUE
2. TRANSMIT PORT A
   DA = LOCAL BROADCAST
   I-Am Request,
   'Device Identifier' = (TD's Device Identifier)
3. WAIT **Internal Processing Fail Time** + 10 seconds
4. VERIFY Device_Address_Proxy_Table = (a list containing the TD)
5. TRANSMIT PORT A
   DA = LOCAL BROADCAST
   DNET = GLOBAL BROADCAST
   DLEN = 0
   I-Am Request,
   'Device Identifier' = (TD's Device Identifier)
6. CHECK (the IUT does not initiate an I-Am request for the TD on Network 2)

© 2026 by BACnet International. All rights reserved.

Version	Date	Author	Change
0.07	5-Aug-2004	Carl Neilson	Updates based on Nashville meeting comments on Round 3 updates.
0.08	24-Aug-2004	Carl Neilson	• Removed 9.24.4.X1, 9.24.4.X2. Now exist in 135.1. • Modified the purpose of 14.5.3. • Modified the purpose of 14.2.2 • Added 10.2.4.4
0.09		Roland Laird	• Modified all Clause 14 tests
0.10	26-Oct-2004	Roland Laird	• Continuation of BACnet/IP modifications - changes highlighted inline
0.11	27-Oct-2004	Carl Neilson	• Added 7.3.1.11, 9.1.1.1, 9.1.1.4, 9.1.2.1, 9.1.2.5. The specification of the expected Time Stamp in the ack notifications was changed. - changes still highlighted inline
0.12	29-Oct-2004	Carl Neilson	• Changes to 7.3.1.11, 9.1.1.1, 9.1.1.4, 9.1.2.1, 9.1.2.5 based on group feedback. • Changes to BACnet/IP based on group feedback
0.13	23-Nov-2004	Carl Neilson	• Added 9.24.1.X2 and 9.24.1.X3 • Added "Reason For Change" to all tests. • Added passing result text to 9.1.2.5 (missed in version 12) • A few minor typos
0.14	20-Dec-2004	Carl Neilson	• Added tests 10.2.2.3, 10.2.2.7.2, 10.2.2.7.3, 10.2.3.2, 10.2.3.5, 10.2.4.6, 10.2.4.8, 10.2.6 from P3-Routing-14. • Modified 10.2.4.4 as per P3-Routing-14. • Added tests 9.20.2.1 into RPM-B • Added 7.3.2.9.8, 7.3.2.9.9, 7.3.2.17.5, 7.3.2.18.6, 7.3.2.19.5, 7.3.2.22.9 into WP-B. • Added 9.23.1.X7, 9.23.1.X8 into WPM-B • Added 7.3.1.X2
0.15	15-Jun-2005	Carl Neilson, Roland Laird	• Modified 7.3.2.9.8's & 7.3.2.9.9's reason for change comments as 135.1a now incorporates the complete change. • Added 8.34.X1 • 13.X.1 fixed test step reference in steps 8 & 18 • 13.4.3 changed 2 < x < 254 to 2 < x <= 254 • Change 9.20.2.1 to 9.20.2.X1 as the test is new and there is already a 9.20.2.1 in 135.1 • Added scheduling tests 7.3.2.22.X2 and 7.3.2.22.X3 from ShedProtRev4Tests-9. Test numbers were changed to correspond with the equivalent pre-revision 4 tests.
0.16	19-Jul-2005	Carl Neilson	• Added WhoHas tests 9.32.1.X1, 9.32.1.X2
0.17	05-Oct-2005	Jim Butler Carl Neilson	• Added Recipient List Test 7.3.2.20.3.X1, • Added MS/TP restart tests 2.2.14...2.2.17 • Added RP fallback tests 8.20.Y1.X1, 8.20.Y1.X2 • Added AckAlarm tests 9.1.1.X1, 9.1.1.X2 • Changed 2.2.7 as per CLB-001 • Changed 2.2.6 as per CLB-002 • Changed 2.2.5 as per CLB-003 • Changed 2.2.4 as per CLB-004 • Added changes to 7.3.2.23.5
0.18	24-Oct-2005	Carl Neilson	• Added ARCNET tests & re-arranged section 2. • Added 7.3.1.X3 Array Sizing Test

© 2026 by BACnet International. All rights reserved.

			• Added 13.X2.1 APDU Retry and Timeout
0.19	27-Oct-2005	Carl Neilson	• Removed router qualification tests. • Added reason for change to 13.X2.1 & modified note to tester • Fixed incorrect numbering of BACnet/IP sections • Deleted old comment as end of 7.3.2.20.3.X1
0.20	17-Jan-2006	Carl Neilson	• Added 8.8.1 & 8.8.2 that include transmission of final BACnet-ComplexACK-PDUs
4.0.0	13-Sep-2006	Carl Neilson	• Changed revision numbering
4.0.1	04-Apr-2007	Carl Neilson	• Round 4 changes (excl SCHED)
4.0.2	02-May-2007	Carl Neilson	• Added 9.10.1.X2
4.0.3	11-Jun-2007	Lori Tribble	• Updated document per CRR-0005, CRR-0008, CRR-0009, CRR-0011, CRR-0014 • Updated document per CRR-0015, CRR-0017, CRR-0020 • Updated document per CRR-0021, CRR0022
4.0.4	23-Jul-2007	Lori Tribble	• Updated the Reason For Change. • Highlighted new items for Round 4 in Green • Highlighted items to be deleted in Yellow. Waiting on approval of round 3 documents before we delete. • Highlighted items with questions in Purple. Waiting on approval of round 3 documents before changing. • See BTL Specified Tests 3.1.4 change log for details.
4.0.5	10-Oct-2007	Lori Tribble	• Removed tests previously highlighted in yellow. These tests are now in 135.1. • Added changes to tests 7.3.2.22.X1.1,2,3,4 per CRR-0030. • Added changes to test 7.3.2.22.X2.3.12 per CRR-0035. • Added changes to tests per WSPLab suggestions.
4.0.6	25-Oct-2007	Lori Tribble	• Removed some of the highlighting. • Updated tests per mtg 10/15/2007
4.0.7	18-Dec-2007	Lori Tribble	• Added Virtual Routing tests • Added List Manipulation Tests
4.0.8	22-Feb-2008	Lori Tribble	• Fixed BTL-7.3.1.11 per TGTC-18.
4.0.9	01-Apr-2008	Lori Tribble	• Updated page header format.
4.0.10	16-Apr-2008	Lori Tribble	• Applied the following: TGTC-05 - Adds UTCTimeSync to all schedule tests. TGTC-08 - Adds 9.1.2.3 and 9.1.2.6 to this document. TGTC-13 - changes already existed in this document. TGTC-14 - Modified 7.3.2.23.9, added 7.3.2.23.10, modified 7.3.2.23.X2. TGTC-16 - Modified 7.3.1.13 TGTC-17 - 7.3.2.23.10 has already been modified by TGTC-14 TGTC-18 - Added 7.3.1.10.X1, TGTC-19 - Modified 9.21.1.4 TGTC-21 - Modified 7.3.2.22.X1.2 and 7.3.2.22.X1.4 TGTC-35 - Added 7.3.1.3. TGTC-36 - Added 7.3.1.10 TGTC-37 - Modified 7.3.1.11 TGTC-40 - Added 9.22.2.4 TGTC-41 - Modified 7.3.2.23.6.1 TGTC-43 - Added 8.22.1, 8.22.2. Modified 8.22.X2.

© 2026 by BACnet International. All rights reserved.

			BTL-CRR-0018 - Modified 9.10.2.1 BTL-CRR-0050 - Modified 7.2.1.10 BTL-CRR-0051 - Modified 13.X1.3, 13.X1.6, 13.X1.7 • Updated reason for change on several tests. • Updated tests for Rev 5 and 6 Added 9.1.1.X3 Added 9.1.2.3, 9.1.2.4, 9.1.2.7 • Added 13.X5.1, 13.X5.2, 13.X5.3, 13.X5.5, 13.X5.6 for Backup and Restore Initiation testing.
4.0.11	April 16, 2008	Lori Tribble	• Accepted Changes made above • Updated table of contents • Added Reason for Change to tests that did not have it. • Marked test 9.10.2.1 for further review
4.0.13	May 21, 2008	Lori Tribble	• Added test correction for 9.33.2.3 per BTL-CRR-0055. • Added test corrections for 9.14.2.3 and 9.15.2.2 per WS-038-4. • Added test corrections for 8.4.1, 8.4.2, 8.4.3.1, 8.4.3.2, 8.4.4, 8.4.5, 8.4.6 per BTL-CRR-0017. • Corrected reason for change on several tests. • Removed : from test numbers
4.0.14	June 20, 2008	Lori Tribble	• Updated tests 7.3.2.22.X1 and 7.3.2.22.X2.3.1 to match recent changes made in TI-WG on SED-004 and SED-006. • Updated all tests which use the UTCTimeSynchronization service to indicate using a UTC date. • Updated non-router tests per CN-092-04 • Question about test 8.4.6 correction to be answered. See comment.
4.0.15	September 9, 2008	Lori Tribble	• Applied BTL-CRR-0056 Time Master changes • Applied BTL-CRR-0064 NonRouterNetworkCommands • Updated document to reference 135.1-2007 section numbers. • Added 7.3.2.8.1 and 7.3.2.8.3. These required updates for UTCTimeSynchronization. • Added 7.3.2.21.3.1 and 7.3.2.21.3.2. These required updates to include UTCTimeSynchronization. • Added 7.3.2.23.1 and 7.3.2.23.2. These required updates to include UTCTimeSynchronization. • Added 7.3.2.23.3.1 - 9 and 7.3.2.23.4 - 8. These tests required updates to include UTCTimeSynchronization.
4.0.16	September 17, 2008	Lori Tribble	• Accepted all changes made previously. • Made format changes.
4.0.18	Oct 21, 2008	Lori Tribble	• Updated Reason For Change for all tests that now have SSPC proposals. • Added client side schedule tests • Removed test 9.23.1.X • Changed COV test from 24 hour lifetime to 8 hour lifetime.
5.0.1	Oct 21, 2008	Lori Tribble	• Accepted all changes made above. Changed version to 5.0.1 • Updated Reason For Change for tests that now have SSPC proposals.

© 2026 by BACnet International. All rights reserved.

			•
5.0.2	Feb 24, 2009	Lori Tribble	• Added place holder for new test BTL-8.22.X4 Writing Array properties as a Whole array. • Renumbered test steps for 7.3.2.21.3.2 • TGTC-57: Updated Configuration Requirements for test 7.3.2.23.X2.3.12 Revision 4 Lower Event Priority Change Test. • TGTC-58: Updated test 7.3.2.23.X2.3.10 Revision 4 Calendar Entry WeekNDay Odd-Numbered Month Test. • TGTC-59: Updated test 7.3.2.24.1 Log_Enable Test. • TGTC-60: Updated test 8.4.2 CHANGE_OF_STATE Tests • TGTC-79: Updated tests 8.2.1 through 8.2.8 to include BEFORE Notification Fail Time before each notification. • TGTC-80: added tests 9.10.1.1 through 9.10.1.3 to wait Notification Fail Time before each notification. • TGTC-81: Added test 7.2.2. • TGTC-84: Updated test 7.3.2.24.10 Notification_Threshold Test • TGTC-85: Updated test 8.4.7 BUFFER_READY Tests • BDS-001: Updated tests 7.3.2.23.5 Exception_Schedule Restoration Test and 7.3.2.23.6 Weekly_Schedule Restoration Test • BTL-CRR-0069: Updated test 10.X.1 Static Router Binding, 10.X.2 Router Binding via Application Layer Services, 10.X.3 Router Binding via Who-Is-Router-To-Network, and 10.X.4 Router Binding via Broadcast. • BTL-CRR-JN3: Updated test 2.2.7. • Added database_revision tests from 135.1-2007f.
5.0.4	27-Mar-2009	Lori Tribble	• Changed test 9.2.1.X8 to be 9.3.X9. The test doesn't exist yet but is supposed to be the unconfirmed version of the 9.2.1.X4 test which is also not written. • Added place holder for 8.4.X2 Extended Algorithm Tests (ConfirmedEventNotification) and 8.5.X3 Extended Algorithm Tests (UnconfirmedEventNotification). • Renumbered 9.23.2.X7 to 9.23.2.6 (as defined in 135.1-2007) • Renumbered 9.23.1.X8 to 9.23.2.7 (as defined in 135.1-2007) • Updated tests 9.14.2.3 and 9.23.2.6 per BTL-CRR-0072 • Updated tests 9.1.1.1 and 9.1.1.4 per TGTC-111 • Updated test 7.3.2.23.X2.3.9 per TGTC-127 • Updated test 7.3.2.21.3.X per TGTC-128 • Updated test 9.22.1.X2 per TGTC-133 • Added test 7.3.2.24.8 per BTL-CRR-0070 • Added Chapter 6 sections that are changed or new to 135.1 and whose contents are being used within some of the tests (i.e. READ) (Described in CN-093) • Added section 7.2.1.3 to document to show proposed change to text. (FR-??)
5.0.5	6-Apr-2009	Lori Tribble	• Added reason for change to chapter 6 section and for test 7.2.2.

© 2026 by BACnet International. All rights reserved.

			• Modified text for 7.2.1.3 and added reason for change. • Added reason for change for the Record_Count test (7.3.2.24.8)
5.0.6	9-Apr-2009	Lori Tribble	• Added test 9.2.20.1 Reading a Single, Unsupported Property from a Single Object. Per CRR-0039. • Fixed spelling error in Configuration Requirements of Stop_When_Full TRUE Test (7.3.2.24.6.1). • Updated Create and Delete Tests per proposal provided to BTL-WG and approved on 4/9/2009. Tests modified are: 8.16.2, 8.16.3, 8.16.4, 9.16.1.1, 9.16.1.2, 9.16.1.3, 9.16.1.4, 9.16.2.1, 9.16.2.2, 9.16.2.3, 9.16.2.4, 9.16.2.5, 9.16.2.6, 9.17.1.1. Also removed test 9.16.1.X1 per this document.
5.0.7	8-Jun-2009	Lori Tribble	• Added to 9.20.2.1 that this change is included in CN-121. • Changed test 7.3.1.1 per BTL-CRR-0074 and DJH-001-3.
5.0.8	22-Jun-2009	Lori Tribble	• Removed test 9.23.1.7 Writing Maximum Multiple Properties test. • Updated test 7.3.1.11 to update the configuration requirements to include initial configuration of the ACK_Required property. • Test 7.3.2.23.X1.1 - updated configuration requirements • Updated tests 7.3.2.23.7, 7.3.2.23.8, 7.3.23.X2.8, 7.3.23.X2.7 step 1 to correctly reference D_t not D_1 • Updated test 7.3.1.10 configuration requirements. Changed 'read-only' to 'not configurable'. • Removed all highlighting • Updated TOC.
5.0.final	26-Jun-2009	Lori Tribble	• Accepted all changes per acceptance by BTL-WG 6/18/2008.
6.0.1	26-Jan-2011	Duffy O'Craven	• Corrected: 'inside' for 'outside' in step 4 of test 7.3.2.8.2, based upon BTL-CRR-0172_7.3.2.8.2_inside_outside.doc • Put section 7.3.2.10 in order, before 7.3.2.21 • Revised test 7.3.2.23.X2.4 Revision 4 Weekly_Schedule and Exception_Schedule Interaction Test , based upon KV-001-03_7.3.2.23.X2.4.doc • Adjusted heading on test instead of section, so that test 7.3.2.24.6.1 appears in Table of Contents. • Added tests 9.1.1.X4 and 9.1.1.X5 to ACK-B, based upon BTL Specified Tests-Add135-2004m-4-ReAckAlarms-3.doc • Removed test 9.1.2.6, as the correct version is now in 135.1-2009, per BTL-CRR-0125_9.1.2.6.doc • Added test 9.21.1.X5 Reading Items with Negative Count and MOREITEMS • Derived tests from 135.1-2009 in DCC-A and RD-A, adding proper password treatment based upon BTL-CRR-0078 DeviceCommunicationControl_Password.doc • Revised tests 9.24.21, 9.24.2.2, 9.27.1.1 and 9.27.1.3, and added tests 9.24.2.X3, 9.27.2.X3 and 9.27.2.X4 in

© 2026 by BACnet International. All rights reserved.

			DCC-B and RD-B, based upon 135-2004m-8 r2 Clarify DeviceCommunicationControll and ReinitializeDevice interactions.doc
9.0.3	7-Apr-2011	Duffy O'Craven	• Incorporated BTL - 7.3-MO_V9.doc including new test 7.3.2.24.X7 • Derived BTL – 7.3.1.12 with modifications in consequence of BTL-CRR-0171_7.3.1.12_TO-NORMAL.doc • Incorporated changes to genericize tests for logging objects in BTL - 8.21-MO V8.doc and BTL - 9.21-MO V7.doc • Incorporated DO-016-08_Verify_Notification_Logging.doc as tests 7.3.2.26.X1, 7.3.2.26.X2, 7.3.2.26.X3, and 7.3.2.26.X4 • Incorporated 135-2004b-5 - Restart Parameters_v2.doc in test 8.3.X1
9.0.4	26-May-2011	Duffy O'Craven	• Incorporated BTL-CRR-0082_ReadOnlyTest.doc in test 7.2.2.1 • Incorporated BTL-CRR-0083_nonDocumentedProperty.doc in test 7.2.2.X2 • Added test 2.2.18 Verify Tno_token w/ Serial Analyzer in consequence of BTL-CRR-0085-NewMSTPTest.doc • Specified Protocol_Revision ≥ 7 in test 13.2 in consequence of BTL-CRR-0087_TimeMasterTest.doc • Added modified test 9.7.1.1, in consequence of BTL-CRR-0089_9.7.1.1.doc
9.0.5	31-May-2011	Duffy O'Craven	• Modified tests 8.2.2, 8.2.4, 8.2.6, and 8.2.8 in consequence of BTL-CRR-0095_changeable_Status_Flags.doc • Modified tests 8.4.4, 8.4.5, 8.4.6, and 8.4.7 in consequence of BTL-CRR-0096_object_referenced_by_EE.doc • Specified Protocol_Revision < 10, in consequence of BTL-CRR-0104_correcting_9.10.2.1.doc • Corrected Test Concept: from TD to IUT in 10.X.3 in consequence of BTL-CRR-0106_10.X.3_TestingHints.doc • Corrected Test Configuration: of test 7.3.2.24.4 in consequence of BTL-CRR-0116_Log_Interval_read-only.doc • Corrected expected result in test 9.14.2.2 in consequence of BTL-CRR-0117_9.14.2.2_First_Failed_Element.doc • Corrected the name of test 9.30.1.1, and added BTL Specified Tests versions of 9.30.1.2, 9.31.1.1 and 9.31.1.2 derived from 135.1 - 2007 as specified in BTL-CRR-0113_9.31.1.1_diverge_dissimilar_tests.doc
9.0.6	09-Jun-2011	Duffy O'Craven	• Added tests 7.3.2.29.X1 and 7.3.2.29.X2 for Structured View in consequence of Structured View Test Plan v6.doc
9.0.7	12-Jun-2011	Duffy O'Craven	• Revised test 7.2.2.X2 to restrict the test to standard object types, in consequence of BTL-CRR-0130_7.2.2.X2.doc and making it identical to the revision in 135.1-2009n-1

© 2026 by BACnet International. All rights reserved.

			• Further revised test 7.2.2.X2, in consequence of BTL-CRR-0180_P_C_C.doc • Further revised test 7.2.2, in consequence of BTL-CRR-0178_allowed-values_REAL.doc • Modified test 10.X.5 to ensure that the packet actually reaches the IUT, and that the test uses an address which resembles the actual address of IUT, in consequence of BTL-CRR-0138_10.X.5_same_DADR.doc • Removed test 7.3.2.21.3.X, as the version in 135.1-2009g-6 replaces it, in consequence of BTL-CRR-0141_7.3.2.21.3.X_DDB_without_range.doc. • Removed tests 8.22.1 and 8.22.2, as 135.1-2009i-7 ratified the Notes to Tester: addition that had caused these revised tests to supercede the 135.1 - 2003 - 8.22.1 and 135.1 - 2003 - 8.22.2 versions. • Removed test 8.22.X1, as the version in 135.1-2009i-8 replaces it. • Added qualifying language in Notes to Tester: of tests 7.3.2.23.X1.3, 7.3.2.23.X1.4, and 7.3.2.23.X2.8 in consequence of BTL-CRR-0158_Sch_Object_Writes.doc • Revised test 10.X.2 in consequence of BTL-CRR-0149_non-BROADCAST.doc • Removed test 14.1.7 in consequence of BTL-CRR-0152_eliminating_14.1.7.doc • Added to the Configuration Requirements: of test 7.3.2.24.X3, in consequence of BTL-CRR-0151_7.3.2.24.X3.doc • Adds a Notes to Tester: to test 7.3.2.24.X1, in consequence of BTL-CRR-0160_Log-interrupted.doc • Further revised test 7.3.2.24.8 in consequence of BTL-CRR-0169_7.3.2.24.7_Not_all_at_once.doc • Derives with modifications, and adds a Notes to Tester: to create test BTL - 7.3.2.24.12, in consequence of BTL-CRR-0165_7.3.2.24.12.doc • Added ‘Server’ = TRUE, to test 7.1 and derived a BTL Specified Test 13.1.12.1 with that change, in consequence of BTL-CRR-0177_server_in_Abort-PDU.doc • Further revised test 9.1.2.3, and derived test BTL - 9.1.2.6 in consequence of BTL-CRR-0195_9.1.2.3_and_9.1.2.6.doc • Further revised tests 9.10.1.1 and 9.10.1.2, in consequence of BTL-CRR-0182_9.10.1.2.doc • Further revised test 9.10.1.1, and derived test BTL - 9.10.1.7, in consequence of BTL-CRR-0194_ACK_in_9.10.1.1_and_9.10.1.7.doc • Further derived 9.10.1.7, in consequence of BTL-CRR-0184-9.10.1.7.doc and BTL-CRR-0200-9.10.1.7.doc • Removed test 9.21.1.4 as 135.1-2009g-16 replaced it, in consequence of BTL-CRR-0201_9.21.1.4.doc
9.0.8	22-Jun-2011	Duffy O’Craven	• Referred from BTL 7.2.2.X2 to test 7.1.x and from BTL 7.2.2.1 to test 7.2.x in 135.1-2009i-22, which is the first

© 2026 by BACnet International. All rights reserved.

			occurrence of this test in a ratified addendum., but with slightly different content. • Added qualifying language in Notes to Tester: of tests7.3.2.23.8 in consequence of BTL-CRR-0158_Sch_Object_Writes.doc • Changed test numbers for tests 7.3.2.23.X2.1 through 7.3.2.23.X2.8 (now 7.3.2.23.X.1 through 7.3.2.23.X.8), 7.3.2.23.X2.3.1 through 7.3.2.23.X2.3.13 (now 7.3.2.23.X.3.1 through 7.3.2.23.X.3.13), and 7.3.2.23.X1 through 7.3.2.23.X4 (now 7.3.2.23.Y.1 through 7.3.2.23.Y.4) and 7.3.2.23.X3 (now 7.3.2.23.Y) to the 135.1-2009j-17 test number used for BUFFER_READY tests. • Changed test number for 8.5.X1 to the 8.5.7 used in 135.1-2009l. • Added more qualifying language in Notes to Tester: of tests 7.3.2.23.X1.3, 7.3.2.23.X1.4, and 7.3.2.23.8 incorporating from the versions in 135.1-2009g-17, 135.1-2009g-21 and 135.1-2009i-7 • Added mention of the version in 135.1-2009j-14, in test 7.3.2.24.4 The BTL Specified Test takes precedence. • Added references to the version in 135.1-2009i-14, in tests 7.3.2.24.X1, 7.3.2.24.X2, and 7.3.2.24.X3 The BTL Specified Tests take precedence. • Added references to the version in 135.1-2009i-14, in tests 7.3.2.24.X4, 7.3.2.24.X6, and 7.3.2.24.X7 The versions of these tests are identical with those in BTL Specified Tests. • Replaced test 7.3.2.24.5 with exactly the 135.1-2009g-16 version, adjusting only some capitalization typos, with no semantic difference. • Added mention of the version in 135.1-2009j-10, in tests 7.3.2.24.6.1, and 7.3.2.24.6.2 The versions of these tests are quite different. • Replaced tests 7.3.2.24.7, and 7.3.2.24.8 with exactly the 135.1-2009j-13 and 135.1-209j-14 PPR1_DRAFT versions, with no semantic difference. • Added references to the versions in 135.1-2009h-3, in tests 8.2.1 through 8.2.8 • Added reference to the version in 135.1-2009i-3, in test 8.2.x1 which is identical with the version in BTL Specified Tests. • Replaced tests 8.18.1, 8.18.2, 8.18.X1, and 8.18.X2 with exactly the 135.1-2009i-4 versions, which compared with the prior BTL Specified version means adjusting a syntactically incorrect VERIFY to CHECK, with no semantic difference. • Specified (BACnetDeviceObjectPropertyReference– referring to the buffer property of the log object) as the first part of Event_Values in every ConfirmedEventNotification of BUFFER_READY event type.
9.0.9	6-Jul-2011	Duffy O'Craven	• Removed tests 7.3.2.10.X3, 7.3.2.10.X4, 7.3.2.10.X5 as these are identical to the versions in 135.1-2009f-2.

© 2026 by BACnet International. All rights reserved.

			• Renumbered test 7.3.2.10.X6 to 7.3.2.10.X4 to match the , number of the corresponding test which is in 135.1-2009f-2, and which is identical, except for an errata, with the version in BTL Specified Tests.
9.0.10	1-Aug-2011	Duffy O'Craven	• Fixed a type "end" for "and", in test 13.1.X6 • Fixed step number reference in step 14 of tests 9.1.1.X4 and 9.1.1.X5 per BTL-CRR-0214_9.1.1.X4_and_9.1.1.X5.doc
9.0.11	28-Sep-2011	Duffy O'Craven	• Incorporated DO-014-01_TimeMaster.doc as tests 13.2.1 through 13.2.7 • Eliminated Chapter 6 Conventions for Specifying BACnet Conformance Tests, since that content is now completely expressed in 135.1-2009 • Corrected the missing underscore typo in Record_Count in test 7.3.2.24.X7, and renumbered the steps to be consecutive.
9.0.12	30-Sep-2011	Duffy O'Craven	• Deleted tests 7.3.1.11, 9.1.1.1, 9.1.1.4, 9.1.2.1 and 9.1.2.5, as the versions from 135.1-2009f-1take precedence. • Deleted tests 10.2.2.3, 10.2.2.7.2, 10.2.3.2, 10.2.3.5, 10.2.4.4, 10.2.4.6, 10.2.4.8, and 10.2.6 as the versions from 135.1-2009g-3 take precedence. • Deleted tests 9.1.1.X1 and 9.1.1.X2 as the versions in 135.1-2009g-4 take precedence. • Deleted test 12.1.1.9.X1 because it is identical to test 12.1.1.9.X in 135.1-2009g-5 • Deleted tests 9.24.1.X2 and 9.24.1.X3 as the versions in 135.1-2009g-8 with numbers 9.24.1.X1 and 9.24.1.X2 take precedence. • Deleted test 7.3.1.1 as the version in 135.1-2009g-9 takes precedence. • Deleted tests 10.X.1, 10.X.2, 10.X.3, and 10.X.4 as the versions in 135.1-2009g-10 - 10.Y.1, 10.Y.2, 10.Y.3, and 10.Y.4 take precedence. • Deleted tests 10.X.5, 10.X.6, and 10.X.7 as the versions in 135.1-2009g-10 - 10.X.1, 10.X.2, and 10.X.3 take precedence. • Added tests 7.3.2.21.1, 7.3.2.21.3.4, and 8.4.8.14 as the versions in 135.1-2009g-11 only portrayed the intended revision with a context-diff, so the entirety of the revised tests is rendered here. • Modified test 8.4.2 with the change in 135.1-2009g-11 • Deleted tests 8.18.X3, 8.22.X2, and 8.22.X3 as the versions in 135.1-2009g-14 - 8.18.3, 8.22.4, and 8.22.5 take precedence. • Deleted tests 9.4.X1, 9.4.X2, 9.5.X1, and 9.5.X2 as the versions in 135.1-2009g-15 - 9.4.5, 9.4.6, 9.5.1, and 9.5.2 take precedence.
9.0.13	10-Oct-2011	Duffy O'Craven	• Deleted test 7.3.2.24.9 as the version in 135.1-2009g-16 takes precedence. • Deleted tests 7.3.2.23.3.1, 7.3.2.23.X.3.1, 7.3.2.23.X.3.2, 7.3.2.23.X.3.3, 7.3.2.23.X.3.4, 7.3.2.23.X.3.5, 7.3.2.23.X.3.6 as the versions in 135.1-2009g-17 take precedence. Note that the test numbers used in 135.1-

© 2026 by BACnet International. All rights reserved.

			2009g-17 each specify X rather than the X2 used in Test Plan-5.0.final and BTL Specified Test-5.0.final. • Deleted tests 13.X3 and 13.X4 as the 13.X1 and 13.X2 versions in 135.1-2009g-19 take precedence. • Deleted tests 8.3.X1 and 9.3.X8 as the versions 8.3.X and 9.3.1 in 135.1-2009g-20 take precedence. • Deleted tests 7.3.2.23.Y.1, 7.3.2.23.Y.2, 7.3.2.23.Y.3, and 7.3.2.23.Y.4 as the versions in 135.1-2009g-21 take precedence. Note that the test numbers used in 135,1-2009g-21 each specify Y rather than the X1 used in Test Plan-5.0.final and BTL Specified Test-5.0.final. • Corrected COLDSTART to WARMSTART in test 7.3.2.23.5 in accordance with 135.1-2009i-1 • Deleted tests 8.8.1 and 8.8.2 as the versions in 135.1-2009i-5 take precedence. • Deleted tests 8.20.Y1.1 and 8.20.Y1.2 as the versions in 135.1-2009i-6 take precedence.
9.0.14	14-Nov-2011	Duffy O'Craven	• Fixed the number on test 9.16.1.2 (was inadvertently 16.1.1.2 in BTL Specified Tests-5.0.final.) • Put test 13.X6.5.1 in the Table of contents, by giving it Header 4 style. • Removed the Notes to tester: section of test 7.3.1.11 which had had the rest of the test removed in revision 9.0.12. • Separated the Purpose and Test Concept of test 7.3.1.13. • Fixed the indentation of step 14. In test 7.3.1.13 • Removed test 7.3.1.X1 as it is identical to the version in 135.1- 2009d-2 - 7.3.2.10.1 • Added Reason for change (to correct a cut&paste&forgot-to-revise typo in the Test Concept) to test 7.3.2.10.X4 • Added Reason for change (the version in 135.1-2009g-11 only portrays the intended revision with a context-diff, so the entirety of the revised test is rendered here) to test 7.3.2.21.3.4 • Removed test 7.3.1.X2 as it is identical to the version in 135.1- 2009i-15 - 7.3.2.11.X • Removed test 7.3.2.21.X1 as it is identical to the version in 135.1- 2009g-7 - 7.3.2.20.X (note that is the second test in that addenda with that same number, there is another in g-6). • Removed tests 9.1.1.X1 and 9.1.1.X2 as the versions in 135.1-2009g-4 take precedence.
9.0.15	23-Nov-2011	Duffy O'Craven	• Removed test 8.34.X1 as it is identical to the version in 135.1- 2009i-12. • Removed tests 9.1.1.X4 and 9.1.1.X5 as the versions in 135.1-2009i-17 take precedence. • Removed test 9.10.1.X2 as the version in 135.1-2009d-1 - 9.10.X takes precedence. • Added Notes to tester: to tests 9.14.2.2 and 9.14.2.3 in consequence of BTL-CRR-0232_9.14.2.2_addl_error_codes.doc, and also applied Protocol_Revision conditional from the version in 135.1-2009i-10 to test 9.14.2.3.

© 2026 by BACnet International. All rights reserved.

			• Removed test 8.16.2 because the correction has already been applied in 135.1-2007. • Removed tests 8.16.3, 8.16.4, 9.16.1.1, 9.16.1.3, 9.16.2.2, 9.16.2.3, 9.16.2.4, and 9.16.2.5 because the versions in 135.1-2009f-3 take precedence. Note that BTL - 9.16.1.4 is preserved for it contains a more accurate restriction of "...any unique object identifier of a type that is creatable *and an instance number that is creatable"* . • Removed tests 9.21.1.1, 9.21.1.2, 9.21.1.3, 9.21.1.4.X1, 9.21.1.6.X1, 9.21.1.6.X2, 9.21.1.X1, 9.21.1.X2, and 9.21.2.X4 because the versions in 135.1-2009i-14 take precedence. Note that BTL - 9.21.1.X3 is preserved for it contains a more accurate list: "Qualifying tests are: 9.21.1.1, 9.21.1.2, 9.21.1.3, 9.21.1.4, 9.21.1.4.X1, 9.21.1.X1 or 9.21.1.X2." • Removed test 9.23.2.6 as the version in 135.1-2009i-10 takes precedence. • Removed test 9.20.2.1 as the version in 135.1-2009i-11 takes precedence. • Removed tests 13.X3 and 13.X4 as the versions in 135.1-2009g-19 take precedence. • Test WARMSTART with no Password is made 9.27.1.3, in correspondence with 135.1-2007 numbering. • Removed entire Chapter 14, replicated in 135-2009e-1
9.0.final	01-Dec-2011	Duffy O'Craven	• Updated from 9.0.15 to 9.0.final, accepting all change tracking
12.0.1	25-Jul-2012	Lori Tribble	• Applied Errata 9.0 7/19/2012 • Applied Addendum 9.0-a • Applied Addendum 9.0-b • Applied Addendum 9.0-c • Applied Errata 12.0 7/23/2012
12.0.2	02-Aug-2012	Lori Tribble	• Applied Errata-BTL Test Package 9.0 plus addenda 8/02/2012 (includes above Errata which was not published)
12.0.final	02-Aug-2012	Lori Tribble	• Accepted all changes and Changed Name
12.1.1	27-Sept-2013	Lori Tribble	• Applied Addendum 12.0b
12.1.2	27-Sept-2013	Lori Tribble	• Applied Addendum 12.0c
12.1.3	30-Sept-2013	Lori Tribble	• Applied Addendum 12.0d
12.1.4	30-Sept-2013	Lori Tribble	• Applied Addendum 12.0e
12.1.5	1-Oct-2013	Lori Tribble	• Applied Addendum 12.0f
12.1.6	1-Oct-2013	Lori Tribble	• Applied Addendum 12.0g
12.1.7	1-Oct-2013	Lori Tribble	• Applied Errata 9/30/2013
14.0.a	1-Nov-2014	Lori Tribble	• Applied Addendum 12.1a
14.0.b	1-Nov-2014	Lori Tribble	• Applied Addendum 12.1b
14.0.c	1-Nov-2014	Lori Tribble	• Applied Addendum 12.1c
14.0.d	1-Nov-2014	Lori Tribble	• Applied Addendum 12.1d
14.0.e	1-Nov-2014	Lori Tribble	• Applied Addendum 12.1e
14.0.plus_errata	3-Nov-2014	Lori Tribble	• Updated Reason for Change on all remaining tests. • Removed some tests which existed in 135.1-2013.
14.0.final	19-Nov-2014	Duffy O'Craven	• Removed comments, and pdated to 14.0.final without change.
15.0.05	24-Aug-2017	Lori Tribble	• Applied Addenda 14.0b-j plus errata

© 2026 by BACnet International. All rights reserved.

15.0.08	25-Sep-2017	Lori Tribble	• Removed test 8.4.X9.
15.0.11	11-Oct-2017	Lori Tribble	• Applied errata from voting members
15.0.final	11-Oct-2017	Lori Tribble	• Accepted all changes
15.1.1	30-Mar-2018	Lori Tribble	• Applied addenda a, b , c, d, and errata
15.1.2	6-Apr-2018	Lori Tribble	• Accepted all changes
15.1.3	26-Apr-2018	Lori Tribble	• Reformatted almost all tests to meet 135.1 formats, applied errata
15.1.4	1-May-2018	Lori Tribble	• Applied Errata
15.1.5	3-May-2018	Lori Tribble	• Fixed formatting and numbering issues.
15.1.final	1-June-2018	Lori Tribble	• Renamed to final
15.2.1		Lori Tribble	• Applied addenda e
15.2.2		Lori Tribble	• Applied addenda f
15.2.3		Lori Tribble	• Applied addenda g
15.2.4	11-Nov-2018	Lori Tribble	• Removed some highlights
15.2.final	11-Nov-2018	Lori Tribble	• Accepted all changes and updated revision
15.2.5	13-Dec-2018	Lori Tribble	• Reverted back to 15.2.4 due to formatting issues. Accepted all changes again and updated revision and date.
15.2.final2	14-Dec-2018	Lori Tribble	• Updated date and revision.
16.0.1	19-Aug-2019	Lori Tribble	• Converted to docx. Updated TOC.
16.0.2	19-Aug-2019	Lori Tribble	• Applied Addenda h.
16.0.3	26-Aug-2019	Lori Tribble	• Applied Addenda i
16.0.4	27-Sug-2019	Lori Tribble	• Applied Addenda j
16.0.5	28-Aug-2019	Lori Tribble	• Applied Addenda k
16.0.6	29-Aug-2019	Lori Tribble	• Applied Addenda l
16.0.7	29-Aug-2019	Lori Tribble	• Applied Addenda m
16.0.8	29-Aug-2019	Lori Tribble	• Applied Addenda n
16.0.9	30-Aug-2019	Lori Tribble	• Applied Addenda o
16.0.10	30-Aug-2019	Lori Tribble	• Applied Addenda p
16.0.11	30-Aug-2019	Lori Tribble	• Applied Addenda r
16.0.12	30-Aug-2019	Lori Tribble	• Applied Addenda s
16.0.13	31-Aug-2019	Lori Tribble	• Applied Errata
16.0.14	19-Sep-2019	Lori Tribble	• Applied all changes. Formatting changes.
16.0.15	25-Sep-2019	Lori Tribble	• Applied Errata.
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	• Applied Errata, added PR21 and PR22 items, renamed to 16.1
18.0_v1	4-Oct-2020	Lori Tribble	• Updated to version 18.0, applied Add ai, applied Add aj, applied addenda bf, applied addenda bg, applied addenda bj, applied addenda fix1, appled addenda fix2, applied addenda misc1, applied addenda al, applied fix3, applied aq, applied as Applied misc3 and misc4 (under misc3 author)
18.0_v2	10-Oct-2020	Lori Tribble	• Applied addenda misc2
18.0_v3	11-Oct-2020	Lori Tribble	• Applied Errata
18_v4	13-Oct-2020	Lori Tribble	• Applied Misc5
18_v5	13-Oct-2020	Lori Tribble	• Applied BTLWG-264
18_v6	18-Oct-2020	Lori Tribble	• Applied BTLWG-1025 • Moved new Notes to Tester to above Test Steps. Left existing to be handled via errata in 135.1.

© 2026 by BACnet International. All rights reserved.

			• Fixed Error Code and Error Class to have quotes around them. • Fixed numbering of some tests.
18.0_v7	29-Oct-2020	Lori Tribble	• Final cleanup after voter comments.
18.1_v1	3-Jan-2021	Lori Tribble	• Creation of document from 18.0_v7
18.1_v4	25-Jan-2021	Lori Tribble	• Added tests from Interim document.
18.1_v5	8-Feb-2021	Lori Tribble	• Added missing WriteGroup tests from Interim document.
18.1_v6	13-Feb-2021	Lori Tribble	• Applied final vote comments.
20.0 v1	18-Nov-2021	Lori Tribble	• Applied fix addenda
20.0 v2	19-Nov-2021	Lori Tribble	• Applied alm addenda
20.0 v3	21-Nov-2021	Lori Tribble	• Applied bc addenda (no changes needed)
20.0 v4	11-Dec-2021	Lori Tribble	• Applied bi addenda
20.0 v5	11-Dec-2021	Lori Tribble	• Applied bk addenda
20.0 v6	11-Dec-2021	Lori Tribble	• Applied cov addenda
20.0 v7	12-Dec-2021	Lori Tribble	• Applied misc1 addenda (no changes)
20.0 v8	12-Dec-2021	Lori Tribble	• Applied log addenda
20.0 v9	15-Dec-2021	Lori tribble	• Applied PR13-PR18 addenda
20.0 v10	15-Dec-2021	Lori Tribble	• Applied bd addenda
20.0 v11	16-Dec-2021	Lori Tribble	• Applied Errata (Part 1)
20.0 v15	02-Jan-2022	Lori Tribble	• Applied comments from voters
20.0 v16	14-Jan-2022	Carl Neilson	• Errata
20.0.1 draft2	25-Apr-2022	Lori Tribble	• Applied crs1 to 20.0.1 draft1 provided by Carl
20.0.1 draft3	1-May-2022	Lori Tribble	• Applied Errata to 20.0.1 and updated TOC.
20.0.1 draft4	7-May-2022	Lori Tribble	• Applied Errata
20.0.1 draft5	19-May-2022	Lori Tribble	• Applied Errata BTLWG-1271
20.0.1 draft6	13-Jun-2022	Lori Tribble	• Applied comments from voting members
20.0.1.draft7	15-Jun-2022	Carl Neilson	• Applied TP validation fixes
20.0.1 final	16-Jun-2022	Emily Hayes	• Final
23.0 v1	25-Oct-2022	Lori Tribble	• Applied vote1 (af, bc, bl, br) changes.
23.0 v2	1-Nov-2022	Lori Tribble	• Applied vote2 (bq, bs, bt, bu, bz) changes.
23.0 v3	1-Nov-2022	Lori Tribble	• Applied vote3 (cd, improvements1) changes.
23.0 v4	4-Nov-2022	Lori Tribble	• Applied vote4 (cr1 and cr2) changes
23.0 v5	11-Nov-2022	Lori Tribble	• Applied Errata as of 11/11/2022
23.0 v6		Carl Neilson	• Validation process before formal vote.
23.0 v7	21-Dev-2022	Lori Tribble	• Applied comments from formal vote.
23.0.1 v1	24-Feb-2023	Lori Tribble	• Applied Errata to 23.0.final
23.1 v2	23-May-2023	Lori Tribble	• Applied cr1, fix, errata
23.1 v3	14-Jun-2023	Lori Tribble	• Corrected TOC entries for several tests
23.1 final	15-Jun-2023	Emily Hayes	• Final
23.2	21-Dec-2023	Michael Osborne	• Add 135.1-2023
23.3	12-Feb-2024	Michael Osborne	• Applied 23.1 fix addendum
26.0 v1	7-Oct-2024	Lori Tribble	• Initial version for 26.0 based on 23.3 v4 with fix1, fix2, fix3, fix4 and epics added.
26.0 v2	9-Oct-2024	Lori Tribble	• Applied cr2
26.0 v3	17-Oct-2024	Lori Tribble	• Applied bq, br, bv, ca, cc, cf
26.0 v4	20-Oct-2024	Lori Tribble	• Applied imp1 and imp2
26.0 v5	23-Oct-2024	Lori Tribble	• Applied imp3 and imp4
26.0 v6	23-Oct-2024	Lori Tribble	• Applied cr2

© 2026 by BACnet International. All rights reserved.

26.0 v7	24-Oct-2024	Michael Osborne	• Added errata • Removed extra returns • Formatted some tests
26.1 v1	3-Sep-2025	Lori Tribble	• Applied TP26.0 Fix1
26.1 v2	4-Sep-2025	Lori Tribble	• Applied TP26.0 Fix2
26.1 v3	24-Sep-2025	Lori Tribble	• Applied TP26.0 Fix3
26.1 v5	25-Sep-2025	Lori Tribble	• Applied TP26.0 cr1 (there is no v4 of this document).
26.1 v6	26-Sep-2025	Lori Tribble	• Applied TP26.0 imp1
26.1 v7	22-Oct-2025	Lori Tribble	• Final Tweaks
26.1 draft2	8-Nov-2025	Michael Osborne	• Final, post-vote
26.1 final	19-Nov-2025	Emily Hayes	• Prep for publication
27.0 v0	June 15, 2026	Michael Osborne	• Initial changes
27.0 v14	8-Oct-2026	Nate Benes	• Applied 26.1 addenda
27.0 v15	1-Nov-2026	Nate Benes	• Draft 15: review comments, WG errata decisions and pre-publication corrections
27.0 final	2-Nov-2026	Emily Hayes	• Prep for publication

© 2026 by BACnet International. All rights reserved.