BACnet® TESTING LABORATORIES ADDENDA

Addendum cr-fix1 to BTL Test Package 26.1

Revision v4
Revised 4/12/2026

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

[This foreword and the "Overview" on the following pages are not part of this Test Package. They are merely informative and do not contain requirements necessary for conformance to the Test Package.]

FOREWORD

The purpose of this addendum is to present current changes being made to the BTL Test Package. These modifications are the result of change proposals made pursuant to the continuous maintenance procedures and of deliberations within the BTL-WG Committee. The changes are summarized below.

In the following document, language to be added to existing clauses within the BTL Test Package 26.1 is indicated through the use of *italics*, while deletions are indicated by ~~strikethrough~~. Where entirely new subclauses are proposed to be added, plain type is used throughout.

In contrast, changes to BTL Specified Tests also contain a yellow highlight to indicate the changes made by this addendum. When this addendum is applied, all highlighting will be removed. Change markings on tests will remain to indicate the difference between the new test and an existing 135.1 test. If a test being modified has never existed in 135.1, the applied result should not contain any change markings. When this is the case, square brackets will be used to describe the changes required for this test.

Each addendum can stand independently unless specifically noted via dependency within the addendum. If multiple addenda change the same test or section, each future released addendum that changes the same test or section will note in square brackets whether or not those changes are reflected.

BTL-26.1 cr-fix1-1: Update Tests that Accept INVALID_TAG [BTLWG-1740, CR-0584]

Overview:

ASHRAE Standard 135 allows a device to use any of the below reject reasons in response to a confirmed request that contains an invalid tag.

- INVALID_TAG,
- INCONSISTENT_PARAMETERS,
- INVALID_PARAMETER_DATA_TYPE,
- MISSING_REQUIRED_PARAMETER, and
- TOO_MANY_ARGUMENTS

SSPC Interpretation from the 2025 fall meeting allows for any reject message.

Changes:

Checklist Changes

None

Test Plan Changes

[Test Plan 4.4.1 change 135.1-2025 - 9.20.2.3 to BTL - 9.20.2.3]
[Test Plan 4.6.1 change 135.1-2025 - 9.22.2.3 to BTL - 9.22.2.3]
[Test Plan 4.8.1 change 135.1-2025 - 9.23.2.3 to BTL - 9.23.2.3]
[All occurrences of test 9.23.2.12 from 135.1-2025 BTL]
[All occurrences of test 9.23.2.19 from 135.1-2025 BTL]
[All occurrences of test 9.23.2.20 from 135.1-2025 BTL]
[All occurrences of test 9.23.2.22 from 135.1-2025 BTL]
[Test Plan 4.8.1 change 135.1-2025 - 9.23.2.3 to BTL - 9.23.2.3]
[Test Plan 8.22.2 change 135.1-2025 - 9.16.2.4 to BTL - 9.16.2.4]
[Test Plan 8.22.3 change 135.1-2025 - 9.16.2.5 to BTL - 9.16.2.5]
[Test Plan 8.24.1 change 135.1-2025 - 9.14.2.2 to BTL - 9.14.2.2]
[Test Plan 8.24.1 change 135.1-2025 - 9.14.2.3 to BTL - 9.14.2.3]

Specified Test Changes

[Add the following tests from 135.1-2025 into BTL Specified Tests and modify as specified]

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.

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~~
2. 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,~~
~~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. VERIFY (L), ListProp = InitialList

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:

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)~~
5. READ X2 =Object_List
6. VERIFY X1 = X2
~~4. CHECK(Verify that the new object was not created)~~
7. 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~~
8. 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)~~
9. READ X2 = Object_List
10. 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.

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)~~

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)~~

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

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' = ROPERTY_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~~
~~(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)~~

9.22.2.3 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,
   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

9.23.2.6 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.

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) |
~~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))
4. IF (a WritePropertyMultiple-Error was received in step 3) THEN
5. VERIFY (O1), P1 = NewValue
ELSE -- a Reject-PDU was received
6. VERIFY (O1), P1 = OldValue

[Modify existing BTL Specified Test with the following changes]

9.23.2.16 WritePropertyMultiple Reject Test for First Element of 'List of Write Access Specifications'

Reason for Change: *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.

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*
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))
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))
4. VERIFY (O1), P1 = X
5. VERIFY (O2), P2 = Y

[Add the following 2 tests from 135.1-2025 into BTL Specified Tests and modify as specified]

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.~~

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))
}

[Modify existing BTL Specified Test with the following changes]

9.23.2.21 DateTime Non-Pattern Properties Test using WritePropertyMultiple Service

Reason for Change: *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~~
~~'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))
}

[Add the following test from 135.1-2025 into BTL Specified Tests and modify as specified]

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,
   '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))

}

BTL-26.1 cr-fix1-2: Fix Test Router Binding Via Who-Is-Router-To-Network [BTLWG-1744, CR-0588]

Overview:

The test is changed to add a condition for device that supports only unconfirmed request and add allowances for both broadcast and unicast unconfirmed services as the expected messages will look different

Changes:

Checklist Changes

None

Test Plan Changes

None

Specified Test Changes

[Modify existing BTL Specified Test]

10.7.3 Router Binding via Who-Is-Router-To-Network

Reason for Change: Fix test to allow for unconfirmed only devices.

Purpose: To verify that the IUT can 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 ~~confirmed~~ 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 ~~ReadProperty~~ request ~~or any other confirmed request~~) THEN {*

5. MAKE (IUT transmit a ~~ReadProperty~~ request to the D2A device on the remote network)
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,
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,

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,
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)

BTL-26.1 cr-fix1-3: COV Out Of Bounds Lifetime Parameter [BTLWG-1751, CR-0591]

Overview:

9.10.2.4 The Lifetime Parameter is Out of Range

The Test Conditionality states: “If the lifetime parameter accepts values across the full range of unsigned values decodable by the IUT (which must be at least the full range of Unsigned16), then this test shall be skipped.”

This needs to be added to 9.11.2.6 and 9.42.2.8.

Changes:

Checklist Changes

None

Test Plan Changes

4.10 Data Sharing - Change Of Value - B

4.10.1 Base Requirements

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

…		
135.1-2025 - 9.10.2.4 - The Lifetime Parameter is Out of Range		
	Test Conditionality	If the lifetime parameter accepts values across the full range of *Unsigned32* ~~unsigned values decodable by the IUT (which must be at least the full range of (Unsigned16)~~, then this test shall be skipped
	Test Directives	
	Testing Hints	

4.20 Data Sharing - Change Of Value Property - B

4.20.1 Base Requirements

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

…		
135.1-2025 - 9.11.2.6 - The Lifetime Parameter is Out of Range		
	Test Conditionality	~~Must be executed if Protocol_Revision >= 15~~ *If the lifetime parameter accepts values across the full range of Unsigned32, then this test shall be skipped.*
	Test Directives	
	Testing Hints	

4.26 Data Sharing - Change Of Value Multiple - B

4.26.1 Base Requirements

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

…
135.1-2025 - 9.42.2.8 - The Lifetime Parameter is Out Of Range

	Test Conditionality	~~Must be executed~~ *If the lifetime parameter accepts values across the full range of Unsigned32, then this test shall be skipped*
	Test Directives	
	Testing Hints	
…		

Specified Test Changes

None

BTL-26.1 cr-fix1-4: Embedded Object Gateway Offline Detection [BTLWG-1758, CR-0593]

Overview:

Some embedded object gateways may not be able to detect when the non-BACnet device is offline. This can occur if the IUT waits forever for a value from the non-BACnet device and never interrogates it.

Changes:

Checklist Changes

None

Test Plan Changes

11.2Gateway - Embedded Objects - B

11.2.1 Base Requirements

Base requirements must be met by any IUT that claims GW-EO-B.

135.1-2025 - 9.18.1.9 - ReadProperty Service when Non-BACnet Device Offline		
	Test Conditionality	Must be executed *if the IUT can detect the non-BACnet device is offline*.
	Test Directives	The test shall be conducted upon an object which *represents* ~~is representing~~ information arriving through a Gateway.
	Testing Hints	
135.1-2025 - 9.20.1.15 - ReadPropertyMultiple Service when Non-BACnet Device Offline		
	Test Conditionality	*Must be executed if the IUT supports the ReadPropertyMultiple service and can either detect the non-BACnet device is offline.* ~~If IUT does not support ReadPropertyMultiple service, then this test shall be skipped.~~
	Test Directives	The test shall be conducted upon an object which *represents* ~~is representing~~ information arriving through a Gateway.
	Testing Hints	
135.1-2025 - 9.21.1.15 - ReadRange Service when Non-BACnet Device Offline		
	Test Conditionality	*Must be executed if the IUT supports the ReadRange service, supports a list property that maps onto data from a non-BACnet device, and can either detect the non-BACnet device is offline.* ~~If IUT does not support ReadRange service then this test shall be skipped.~~ ~~If IUT supports the ReadRange service but does not support a list property that maps onto data from a non-BACnet device, this test shall be skipped.~~
	Test Directives	The test shall be conducted upon an object which *represents* ~~is representing~~ information arriving through a Gateway.
	Testing Hints	

Specified Test Changes

None

BTL-26.1 cr-fix1-5: Fix Lighting Command Operation RAMP_TO Test [BTLWG-1820, CR-0605]

Overview:

Incorrect In_Progress states in steps 10 and 24.

Changes:

Checklist Changes

None

Test Plan Changes

None

Specified Test Changes

[Update BTL Specified Test 7.3.2.39.5]

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%).

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~~
26. CHECK (Tracking_Value is decreasing with the ramp-rate)
    }

-- Check In_Progress and Tracking_Value
27. VERIFY In_Progress = IDLE
28. VERIFY Tracking_Value = V2

BTL-26.1 cr-fix1-6: Execute Who-Id from a Remote Network [BTLWG-1745, CR-0585]

Overview:

Change Request CR-0585 was raised to clarify whether 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. A note has been added at the title of the test to indicate that IUT is allowed to raise Who-Is-Router-To-Network message.

Changes:

Checklist Changes

None

Test Plan Changes

None

Specified Test Changes

[Move section from 135.1 into BTL Specified Tests and add new paragraph description.]

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.

BTL-26.1 cr-fix1-7: Time_Of_Device_Restart Value Resolution [BTLWG-1833, CR-0608]

Overview:

Work item to resolve CR-0608. Update Test 8.3.10 to allow Time_Of_Device_Restart to be any valid value.

Changes:

Checklist Changes

None

Test Plan Changes

[In Clause 8.20.1, change 135.1-2025 - 8.3.10 to BTL - 8.3.10]

Specified Test Changes

[Add to BTL Specified Test 8.3.10, renumber test steps as appropriate

8.3.10 Device Restart Notifications

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.*

Test Steps:

1. IF (Restart_Notification_Recipients is writable) THEN
2. WRITE(Restart_Notification_Recipients = any non-empty list of Recipients)
   ELSE
3. MAKE (Restart_Notification_Recipients contain any non-empty list of Recipients)
4. MAKE(the IUT reset)
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 = (T2),
Last_Restart_Reason=(any valid restart reason, R))
}
8. VERIFY Time_Of_Device_Restart = T2
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)