Clarification Request

References: BTL Specified Tests version of test 7.2.2, 9.22.1.X2, 9.22.2.1, 9.22.2.3, 9.22.2.4.

Date of BTL-WG Response: February 7, 2013

Background:

BTL Specified Tests versions of tests 7.2.2, 9.22.1.X2, 9.22.2.1, 9.22.2.3, 9.22.2.4:

7.2.2 Write Support Test Procedure

Reason for Change: This test did not account for restrictions placed on properties by the vendor. This is not in any new SSPC proposals.

Purpose: To verify that all writable properties of all objects can be written to using BACnet WriteProperty and WritePropertyMulitiple services. The test is performed once using WriteProperty and once using WritePropertyMultiple. When writing to array properties, the whole array shall be written without using an array index, where possible.

Dependencies: ReadProperty Service Execution Tests, 9.18; WriteProperty Service Execution Tests, 9.22.

Test Steps:

1. REPEAT X = (all objects in the IUT's database) DO {
   REPEAT Y = (all writable properties in object X) DO {
   REPEAT Z = (all values meeting the functional range requirements of 7.2.1*, and any additional restrictions placed on the allowable property values by the vendor using an English language description for their allowed values for any property. Or by the object or property definition*) DO {
   WRITE (X), Y = Z,
   VERIFY (X), Y = Z
   }
   }
   }

When the Setpoint_Reference list contains 1 element, the setpoint for this control loop is the value from a property of another object. Setpoint_Reference specifies that object and property. The Issue raised regarding Setpoint in testing on 16-Jan-2013, is that the lab observed that After transmitting WP request to Setpoint property with valid datatype, IUT yet responded with reject reason: Inconsistent_Parameters.

9.22.1.X2 Writing to Properties Based on Data Type

Reason for Change: A general WriteProperty test is not supplied by 135.1 that can be used in a variety of situations. The BTL-WG has kept this test to ensure that all data types are tested.

Purpose: This test case verifies that the IUT can execute WriteProperty service requests to specific data types supported by the IUT.

Test Concept: For the specified base data type, the TD shall select an object in the IUT that contains a writable property of that data type. This property is designated P1.

Configuration Requirements: The IUT shall be configured with at least one writable property of the specified data type to be used for this test.

Test Steps:

1. X = READ (Object1), P1
2. TRANSMIT WriteProperty-Request,
   'Object Identifier' = Object1,
   'Property Identifier' = P1,
   'Property Value' = (any valid value defined for this property subject to the restrictions specified in the EPICS as defined in 4.4.2, except the value X determined in step 1)
3. RECEIVE Simple-ACK-PDU
4. VERIFY (Object1), P1 = (the value used in step 2)

9.22.2.1 Writing Non-Array Properties with an Array Index

Reason for Change: Modified test to remove dependency on EPICS values.

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. 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

~~1. VERIFY (Object1), P1 = (the value defined for this property in the EPICS)~~

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
   RECEIVE BACnet-Error PDU,
   Error Class = PROPERTY,
   Error Code = PROPERTY_IS_NOT_AN_ARRAY
   ELSE
   RECEIVE BACnet-Error PDU,
   Error Class = SERVICES,
   Error Code = INCONSISTENT_PARAMETERS
4. VERIFY (Object1), P1 = *X*~~(the value defined for this property in the EPICS)~~

9.22.2.3 Writing with a Property Value Having the Wrong Datatype

Reason for Change: Relax Tests of INVALID_DATATYPE per INTERPRETATION IC 135-2004-28. Modified Test to remove dependency on EPICS values.

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 an invalid datatype. If no object supports writable properties, then this test shall be omitted.

Test Steps:

1. READ X = (Object1), P1
~~1. VERIFY (Object1), P1 = (the value defined for this property in the EPICS)~~
2. TRANSMIT WriteProperty-Request,
'Object Identifier' = Object1,
'Property Identifier' = P1,
'Property Value' = (any value with an invalid datatype)
3. ~~IF (Protocol_Revision is present and Protocol_Revision ≥ 4) THEN~~
~~RECEIVE BACnet-Error PDU,~~
~~Error Class = PROPERTY,~~
~~Error Code = INVALID_DATATYPE~~
~~ELSE~~
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)
4. VERIFY (Object1), P1 = *X*~~(the value defined for this property in the EPICS)~~

9.22.2.4 Writing with a Property Value that is Out of Range
Reason for Change: Based on test in 135.1-2009i and then modified test to remove dependency on EPICS values.

Purpose: To verify that the IUT can execute WriteProperty service requests when an attempt is made to write a value that is outside of the supported range.

Test Concept: The TD attempts to write to a property using a value that is outside of the supported range. *If the IUT does not contain any writable properties that have restricted ranges, then this test shall be skipped.*

Test Steps:

1. READ X = (Object1), P1
~~1. VERIFY (Object1), P1 = (the value defined for this property in the EPICS),~~
2. TRANSMIT WriteProperty-Request,
'Object Identifier' = (Object1, any object with writable properties),
'Property Identifier' = (P1, any property with a restricted range of values),
'Property Value' = (any value that is outside the supported range)
1. IF (Protocol_Revision is present and Protocol_Revision >= 4) THEN
RECEIVE BACnet-Error PDU,
Error Class = PROPERTY,
Error Code = VALUE_OUT_OF_RANGE
ELSE
RECEIVE (BACnet-Error-PDU,
Error Class = PROPERTY,
Error Code = VALUE_OUT_OF_RANGE) |

(BACnet-Reject-PDU,
Reject Reason = PARAMETER_OUT_OF_RANGE)

4. VERIFY (Object1), P1 = *X*~~(the value defined for this property in the EPICS)~~

Notes to tester: The value used in step 2 shall be of the correct datatype. For bit string types, the bit count shall be correct, for Date and Time values, the value shall be within the range defined by the standard for the datatype, for constructed values, the constructed value shall match the structure defined by the ASN.1 and all field values shall be within the ranges defined by the standard for those field values.

When the request packet conveys a valid datatype to a sometimes writable property, Reject reason: Inconsistent_Parameters is currently outside of the allowed responses specified in the Test Plan, though I think it fits the situation. I request the BTL-WG relax all the property modifying service tests (in WriteProperty, WritePropertyMultiple, AddListElement, CreateObject, and eventually WriteGroup, to include that possible Reject reason is allowed in this situation.

Question:

Is it allowed for a BACnet server device to return a Reject-PDU with reject reason INCONSISTENT_PARAMETERS as the appropriate error code in cases where accepting the supplied value would create semantic inconsistency?

Response:

No. BACnet-Reject-PDU (see clause 20.1.8) is for syntax or protocol errors, which this is not. Error Class: PROPERTY Error Code: VALUE_OUT_OF_RANGE is the appropriate response in this situation (see clause 15.9.1.3.1