BACnet® TESTING LABORATORIES ADDENDA

Addendum bd to BTL Test Package 18.1

Revision v3
Revised 12/1/2021

Approved by the BTL Working Group on November 11, 2021.
Approved by the BTL Working Group Voting Members on November 30, 2021.
Published on December 7, 2021.

[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 18.1 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-18.1-bd-1: Add Testing for Staging Object [BTLWG-421]

Overview:

Addendum 2016bd (PR20) added support for the Staging Object type. 3 interesting things regarding the Staging Object type:

1) The Staging Object has Stages and Stage_Names properties that have similar functional requirements as a multi-state objects Number_of_State and State_Text.
2) The Staging Object writes to Target_References (internal or external) for which there are similar functional requirements with the Schedule's List_Of_Object_Property_References property. And both objects use the Priority_For_Writing property the same way.
3) Unique interrelationship with Stages and Target_References properties. The Stages property is a complex data type (limit, values, deadband) where 'values' is a bitstring of length N. The bits correspond to Target_References array members so these items must align. Target_References[0] = N

Changes:

BTL Checklist Changes

[Modify the Staging Object section]

Staging Object		
	R[1,2]	Base Requirements
	S	*Supports writable Out_Of_Service property*
	O	*Supports configurable Stages property*
	O	*Supports Stage_Names property*
	O	*Supports writable Target_References property*
	O	*Supports the value source mechanism*
[1]~~Contact BTL for interim tests for this object~~ [2]~~Protocol_Revision 20 or higher must be claimed~~		

[In Data Sharing section, delete/renumber footnotes relevant to Staging objects in COV-A and COV-B]

Data Sharing - Change Of Value - A		
		…
	C[2,4,5]	Can subscribe for COV from Staging objects
		…
[1] At least one of these options is required in order to claim conformance to this BIBB. [2] At least one of these options is required in order to claim conformance to this BIBB. [3] Support for this option is suggested except in the case where the device is able to generate infinite subscriptions in which case it is required. [4]~~Contact BTL for interim tests for this object.~~ [5]~~Protocol_Revision 20 or higher must be claimed.~~		

Data Sharing - Change Of Value - B		
		…
	C[1,2,3]	Supports COV for Staging objects
		…
[1] At least one of these options is required in order to claim conformance to this BIBB. [2]~~Contact BTL for interim tests for this object.~~ [3]~~Protocol_Revision 20 or higher must be claimed.~~		

BTL Test Plan Changes

3.62 Staging Object

[Modify the Staging Object section Base Requirements]

3.62.1 Base Requirements

~~Contact BTL for interim tests for this object~~ *Base requirements must be met by any IUT that can contain Staging objects.*

BTL - 7.3.2.X66.1 - Clamping Present_Value to Max_Pres_Value or Min_Pres_Value		
	Test Conditionality	Must be executed
	Test Directives	
	Testing Hints	
BTL - 7.3.2.X66.2 - Present_Stage Evaluation		
	Test Conditionality	Must be executed
	Test Directives	
	Testing Hints	
BTL - 7.3.2.X66.3 - Present_Stage Evaluates on Restart		
	Test Conditionality	If the IUT does not support remote references and does not support non-zero deadbands, then this test shall be skipped.
	Test Directives	
	Testing Hints	
BTL - 7.3.2.X66.4 - Default_Present_Value is Abided on Restart		
	Test Conditionality	Must be executed
	Test Directives	
	Testing Hints	
BTL - 7.3.2.X66.5 - Writing to Target References		
	Test Conditionality	Must be executed
	Test Directives	
	Testing Hints	
BTL - 7.3.2.X66.6 - Stage Value Bitstring is Same Length as Target_References		
	Test Conditionality	
	Test Directives	
	Testing Hints	
BTL - 7.3.2.X66.7 - Max_Pres_Value Equals Last Stage Limit		
	Test Conditionality	Must be executed
	Test Directives	
	Testing Hints	
BTL - 7.3.2.X66.8 - CONFIGURATION_ERROR when Min_Pres_Value is too Large		
	Test Conditionality	If the Min_Pres_Value and Stages properties are Read-Only, this test shall be skipped.
	Test Directives	
	Testing Hints	
BTL -7.3.2.X66.9 - COMMUNICATION_FAILURE on Failed Write to External Target Reference		
	Test Conditionality	If the IUT cannot be configured with an external object in the Target_References property, this test shall be skipped.
	Test Directives	
	Testing Hints	
BTL -7.3.2.X66.10 - Fault Indicated on Failed Write to Local Target Reference		
	Test Conditionality	If the IUT cannot be configured to reference a non-writable, or non-existent, local target, this test shall be skipped.
	Test Directives	
	Testing Hints	

3.62.2 Supports Writable Out_of_Service Property

The Out_Of_Service property in Staging Objects contained in the IUT is writable

BTL -7.3.2.X66.11 - Out_Of_Service, Status_Flags, and Reliability for Staging Object		
	Test Conditionality	Must be executed

	Test Directives	
	Testing Hints	

3.62.3 Supports Configurable Stages Property

The Stages property in Staging Objects contained in the IUT is configurable.

BTL - 7.3.2.X66.12 - Stages Array Sizing Test		
	Test Conditionality	If the IUT cannot be made to contain a Staging object that is resizable by writing to ARRAY INDEX = 0, this test shall be skipped.
	Test Directives	
	Testing Hints	
BTL - 7.3.2.X66.13 - Present_Stage Evaluates on Change to Stages Property		
	Test Conditionality	Must be executed
	Test Directives	
	Testing Hints	
BTL - 7.3.2.X66.14 - CONFIGURATION_ERROR when Limits are Out of Order		
	Test Conditionality	If the Stages property is not writable and cannot be configured with non-ascending Stages, this test shall be skipped.
	Test Directives	
	Testing Hints	
BTL - 7.3.2.X66.15 - CONFIGURATION_ERROR when Deadband < 0		
	Test Conditionality	If the Stages property is not writable and cannot be configured with a Deadband value < 0, this test shall be skipped.
	Test Directives	
	Testing Hints	
BTL - 7.3.2.X66.16 - CONFIGURATION_ERROR when Stages size is less than Two		
	Test Conditionality	If the Stages property is not writable and cannot be configured such that the size of the array is less than two, this test shall be skipped.
	Test Directives	
	Testing Hints	

3.62.4 Supports Stage_Names Property

At least one Staging Object in the IUT supports the Stage_Names property.

BTL - 7.3.2.X66.17 - Stage_Names and Stages Size Equality Test		
	Test Conditionality	Must be executed
	Test Directives	
	Testing Hints	
BTL - 7.3.2.X66.18 - Stage_Names Array Sizing Test		
	Test Conditionality	If the Stage_Names property cannot be resized by writing to it, this test shall be skipped.
	Test Directives	
	Testing Hints	

3.62.5 Supports Writable Target_References Property

The IUT supports a writable Target_References property.

BTL -7.3.2.X66.19 - Target_References Array Sizing Test		
	Test Conditionality	Must be executed
	Test Directives	
	Testing Hints	
BTL - 7.3.2.X66.20 - Writing Target_References with an Unsupported External Reference		
	Test Conditionality	If the IUT allows external references in the Target_References property, this test shall be skipped.
	Test Directives	
	Testing Hints	

3.62.6 Supports the Value Source Mechanism

The IUT supports the Value Source Mechanism in staging objects.

BTL - 7.3.1.X42.Y2 - Non-commandable Value_Source Property Test		
	Test Conditionality	Must be executed
	Test Directives	
	Testing Hints	
BTL - 7.3.1.X42.Y1 - Writing to the Value_Source Property by a Device Other than the Device that Commanded the Property.		
	Test Conditionality	Must be executed
	Test Directives	
	Testing Hints	

[Modify all references to test 135.1-2019 9.2.1.1 to BTL 9.2.1.1, 135.1-2019 9.3.2 to BTL 9.3.2 in the test plan]

***BTL~~135.1-2019~~* - 9.2.1.1 - Change of Value Notifications**		
	Test Conditionality	Must be executed.
	Test Directives	Test one instance of each object type.
	Testing Hints	
***BTL ~~135.1-2019~~* - 9.3.2 - Change of Value Notifications**		
	Test Conditionality	Must be executed.
	Test Directives	Test one instance of each object type.
	Testing Hints	

[Modify section 4.10.34 Supports COV for Staging Objects in the test plan]

4.10.34 Supports COV for Staging Objects

The IUT can subscribe for, receive, and process Change of Value notifications from Staging Objects.

~~Contact BTL for interim tests for this object.~~

BTL - 8.2.X17 - Change of Value Notification of Staging Object Present_Value property		
	Test Conditionality	If the IUT cannot contain a Staging object where the COV_Increment is less than the Present_Value range for a single stage, this test shall be skipped. This may be skipped if 8.3.X17 is executed against a Staging object.
	Test Directives	The selected object must be a Staging object.
	Testing Hints	
BTL - 8.2.X18 - Change of Value Notification of Staging Object Status_Flags property		
	Test Conditionality	If the IUT cannot contain a Staging object where the Status_Flags property can be changed, this test shall be skipped. This may be skipped if 8.3.X18 is executed against a Staging object.
	Test Directives	The selected object must be a Staging object.
	Testing Hints	
BTL - 8.2.X19 - Change of Value Notification of Staging Object Present_Stage property		
	Test Conditionality	This may be skipped if 8.3.X19 is executed against a Staging object.
	Test Directives	The selected object must be a Staging object.
	Testing Hints	
BTL - 8.3.X17 - Change of Value Notification of Staging Object Present_Value property		
	Test Conditionality	If the IUT cannot contain a staging object where the COV_Increment is less than the Present_Value range for a single stage, this test shall be skipped. This may be skipped if 8.2.X17 is executed against a Staging object.
	Test Directives	The selected object must be a Staging object.
	Testing Hints	
BTL - 8.3.X18 - Change of Value Notification of Staging Object Status_Flags property		
	Test Conditionality	If the IUT cannot contain a Staging object where the Status_Flags property can be changed, this test shall be skipped. This may be skipped if 8.2.X18 is executed against a Staging object.

	Test Directives	The selected object must be a Staging object.
	Testing Hints	
BTL - 8.3.X19 - Change of Value Notification of Staging Object Present_Stage property		
	Test Conditionality	This may be skipped if 8.2.X19 is executed against a Staging object.
	Test Directives	The selected object must be a Staging object.
	Testing Hints	

Test Changes

[Add new tests for the Staging Object in Section 7.3.2]

7.3.2.X66.1 Clamping Present_Value to Max_Pres_Value or Min_Pres_Value
Reason for Change: No test exists for this functionality.

Purpose: To verify that Present_Value will be modified internally to stay within the boundaries of Min_Pres_Value or Max_Pres_Value.

Test Concept: Present_Value is written with a value greater than Max_Pres_Value. If the value is accepted, Present_Value is read to verify that it clamped to Max_Pres_Value. If Stages is writable, an attempt is made to reduce the limit defined in the last stage. If successful, Present_Value is checked to verify it changed to match the new limit. Present_Value is then written with a value less than Min_Pres_Value. If the value is accepted, Present_Value is read to verify that it clamped to Min_Pres_Value. If Min_Pres_Value is writable, the value is increased and Present_Value is read to verify that it matches the new Min_Pres_Value.

Configuration Requirements: None

Test Steps:

1. READ MAXPV1 = Max_Pres_Value
2. READ PV1 = Present_Value
3. TRANSMIT WriteProperty-Request
   'Object-Identifier' = (the Staging object under test),
   'Property Identifer' = Present_Value,
   'Property Value' = (a value greater than MAXPV1)
4. RECEIVE BACnet-SimpleACK-PDU |
   (BACnet-Error-PDU,
   'Error Class' = Property,
   'Error Code' = VALUE_OUT_OF_RANGE)
5. IF (a BACnet-SimpleACK-PDU was received) THEN
   VERIFY Present_Value = MAXPV1
   ELSE
   VERIFY Present_Value = PV1
   WRITE Present_Value = MAXPV1
6. IF (Stages is writable) THEN
   READ NS = Stages[0]
   READ STGN = Stages, ARRAY INDEX = NS
   TRANSMIT WriteProperty-Request
   'Object-Identifier' = (the Staging object under test),
   'Property Identifer' = Stages,
   'Property Array Index' = NS,
   'Property Value' = {
   Limit = (STAGEPV1: any value less than STGN.Limit)
   Values = STGN.Values,
   DeadBand = STGN.Deadband
   }
   RECEIVE BACnet-SimpleACK-PDU |
   (BACnet-Error-PDU,
   'Error Class' = PROPERTY,
   'Error Code' = VALUE_OUT_OF_RANGE)
   IF (a BACnet-SimpleACK-PDU was received) THEN
   VERIFY Present_Value = STAGEPV1
7. READ MINPV1 = Min_Pres_Value
8. READ PV2 = Present_Value
9. TRANSMIT WriteProperty-Request
   'Object-Identifier' = (the Staging object under test),
   'Property Identifer' = Present_Value,

'Property Value' = (a value less than MINPV1)

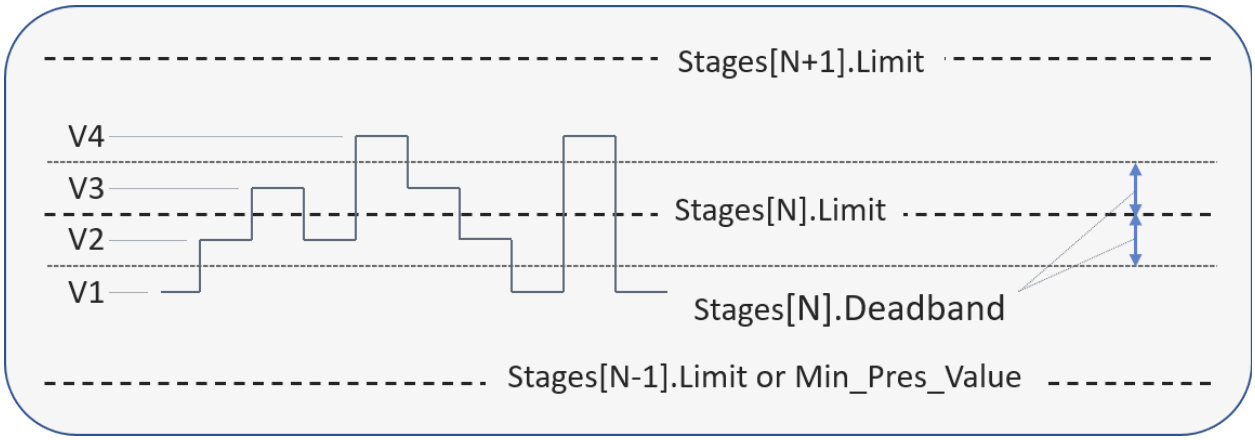
10. RECEIVE BACnet-SimpleACK-PDU |
    (BACnet-Error-PDU,
    'Error Class' = PROPERTY,
    'Error Code' = VALUE_OUT_OF_RANGE)
11. IF (a BACnet-SimpleACK-PDU was received) THEN
    VERIFY Present_Value = MINPV1
    ELSE
    VERIFY Present_Value = PV2
    WRITE Present_Value = MINPV1
12. IF (Min_Pres_Value is writable) THEN
    READ STG1 = Stages, ARRAY INDEX = 1
    TRANSMIT WriteProperty-Request
    'Object-Identifier' = (the Staging object under test),
    'Property Identifer' = Min_Pres_Value,
    'Property Value' = (MINPV2: MINPV1< MINPV2< (STG1.Limit - STG1.Deadband))
    WAIT **Internal Processing Fail Time**
    VERIFY Present_Value = MINPV2

7.3.2.X66.2 Present_Stage Evaluation

Reason for Change: No test exists for this functionality.

Purpose: To verify that Present_Stage evaluates correctly based on the Present_Value, stage limits, and deadband values.

Test Concept: Present_Value is written with different values that exercise the Present_Stage evaluation algorithm. After each write to Present_Value, Present_Stage is read to verify that the algorithm evaluates correctly.



Configuration Requirements: The Staging object used for this test must be configured with at least two stages. If supported, Deadband shall be configured with a non-zero value for stage N (Stage[N].deadband <> 0). At the start of the test, the Staging object is configured with Present_Value = V1.

Test Steps:

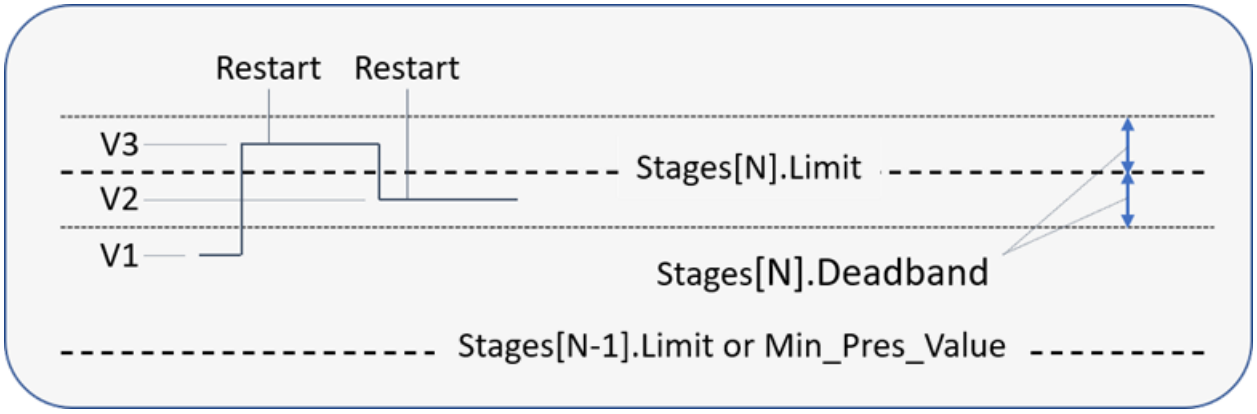
1. READ N = Present_Stage
2. VERIFY Present_Value = V1
3. If (Stages[N].Deadband > 0) THEN {
    WRITE Present_Value = (V2: Stages[N].Limit - Stages[N].Deadband < V2 < Stages[N].Limit)
    VERIFY Present_Stage = N
    WRITE Present_Value = (V3: Stages[N].Limit < V3 < Stages[N].Limit + Stages[N].Deadband)
    VERIFY Present_Stage = N
    WRITE Present_Value = V2
    VERIFY Present_Stage = N
    WRITE Present_Value = (V4: Stages[N].Limit + Stages[N].Deadband < V4 < Stages[N+1].Limit)
    VERIFY Present_Stage = N+1
    WRITE Present_Value = V3
    VERIFY Present_Stage = N+1

WRITE Present_Value = V2
VERIFY Present_Stage = N+1
WRITE Present_Value = V1
VERIFY Present_Stage = N
}
4. WRITE Present_Value = V4
5. VERIFY Present_Stage = N+1
6. WRITE Present_Value = V1
7. VERIFY Present_Stage = N

7.3.2.X66.3 Present_Stage Evaluates on Restart
Reason for Change: No test exists for this functionality.

Purpose: To verify that Present_Stage is re-evaluated on device restart.

Test Concept: Present_Value is written with a value, V3, that exceeds Stages[N].limit but does not exceed the deadband threshold and cause a change to Present_Stage. The IUT is restarted and Present_Stage is read to verify that it is now (N+1). Present_Value is then written with a value, V2, that is below Stages[N].limit but above the deadband threshold so Present_Stage remains at (N+1). The IUT is restarted and Present_Stage is read to verify that it is now N.



Configuration Requirements: The Staging object used for this test must be configured with at least two stages. Deadband shall be configured with a non-zero value for stage N (Stage[N].Deadband <> 0). If deadband for stage N cannot be configured this way in a Staging object which does not support Default_Present_Value, this test shall be skipped. At the start of the test, the Staging object is configured with Present_Value = V1 and Present_Stage = N. If the IUT supports remote Target_References then at least 1 shall be set to an object outside the IUT.

Test Steps:

1. VERIFY Present_Stage = N
2. VERIFY Present_Value = V1
3. WRITE Present_Value = (V3: Stages[N].Limit < V3 < (Stages[N].Limit + Stages[N].Deadband))
4. VERIFY Present_Stage = N
5. IF (ReinitializeDevice - WARMSTART execution is supported) THEN {
   TRANSMIT ReinitializeDevice-Request,
   'Reinitialized State of Device' = WARMSTART
   'Password' = (any valid password)
   RECEIVE BACnet-SimpleACK-PDU
   } ELSE {
   MAKE (power cycle the IUT to make it reinitialize)
   }
6. WAIT for the IUT to complete its restart
7. CHECK(that the IUT wrote to all Target References which are outside the device)
8. VERIFY Present_Value = V3
9. VERIFY Present_Stage = N+1
10. WRITE Present_Value = (V2: (Stages[N].Limit - Stages[N].Deadband) < V2 < Stages[N].Limit)
11. VERIFY Present_Stage = N+1
12. IF (ReinitializeDevice - WARMSTART execution is supported) THEN {
    TRANSMIT ReinitializeDevice-Request,
    'Reinitialized State of Device' = WARMSTART

'Password' = (any valid password)
RECEIVE BACnet-SimpleACK-PDU
} ELSE {
MAKE (power cycle the IUT to make it reinitialize)
}
13. WAIT for the IUT to complete its restart
14. CHECK(that the IUT wrote to all Target References which are outside the device)
15. VERIFY Present_Value = V2
16. VERIFY Present_Stage = N+1

7.3.2.X66.4 Default_Present_Value is Abided on Restart
Reason for Change: No test exists for this functionality.

Purpose: To verify that Default_Present_Value defines the Staging object's value on device restart.

Test Concept: A staging object which contains Default_Present_Value. The stage associated with Default_Present_Value is S1. The staging object starts with the value V2, which evaluates to a different stage, S2. The IUT is restarted and it is verified that the staging object takes on Default_Present_Value, changes to the stage S1 and performs the associated writes. The IUT is restarted again and it is verified that the staging object maintains its value, remains in stage S1 and performs the associated writes for the stage S1.

Configuration Requirements: If the IUT supports remote Target_References then at least 1 shall be set to an object in the TD.

Test Steps:

1. VERIFY Default_Present_Value = V1
2. VERIFY Present_Value = V2
3. VERIFY Present_Stage = S2
4. IF (ReinitializeDevice - WARMSTART execution is supported) THEN {
TRANSMIT ReinitializeDevice-Request,
'Reinitialized State of Device' = WARMSTART
'Password' = (any valid password)
RECEIVE BACnet-SimpleACK-PDU
} ELSE {
MAKE (power cycle the IUT to make it reinitialize)
}
5. WAIT for the IUT to complete its restart
6. CHECK(that the IUT wrote to all Target References which are outside the device)
7. VERIFY Present_Value = V1
8. VERIFY Present_Stage = S1
9. IF (ReinitializeDevice - WARMSTART execution is supported) THEN {
TRANSMIT ReinitializeDevice-Request,
'Reinitialized State of Device' = WARMSTART
'Password' = (any valid password)
RECEIVE BACnet-SimpleACK-PDU
} ELSE {
MAKE (power cycle the IUT to make it reinitialize)
}
10. WAIT for the IUT to complete its restart
11. CHECK(that the IUT wrote to all Target References which are outside the device)
12. VERIFY Present_Value = V1
13. VERIFY Present_Stage = S1

7.3.2.X66.5 Writing to Target References
Reason for Change: No test exists for this functionality.

Purpose: To verify that a change of Present_Stage results in the target references being written as per the stage definition.

Test Concept: A Stage object, O1, is selected for testing. O1's Present_Value is written with a value that results in a change of Present_Stage. Each Present_Value of the Target_References is monitored to verify that its value is set in accordance with Stage[Present_Stage].Values. O1's Present_Value is written again with a value that returns Present_Stage to its initial value. Again, the Target_References are monitored to verify that they have been written with the appropriate values.

Configuration Requirements: Target_References is configured with references to existing binary objects with writable Present_Value properties. The Stages property is configured with at least two stages, X and Y, such that Stages[X].Values <> Stages[Y].Values. Present_Stage shall be X at the start of the test. Throughout the test, O1 is expected to be properly configured such that Reliability is NO_FAULT_DETECTED.

Test Steps:

1. VERIFY Present_Stage = X
2. WRITE Present_Value = (any value that causes Present_Stage to change to Y)
3. VERIFY Present_Stage = Y
4. REPEAT J = (1 ... Target_References[0]) = DO {
   READ O = Target_References, ARRAY INDEX = J
   VERIFY O, Present_Value = Stages[Y].Values[J]
5. WRITE Present_Value = (any value that causes Present_Stage to change to X)
6. VERIFY Present_Stage = X
7. REPEAT J = (1.. Target_References[0]) = DO {
   READ O = Target_References, ARRAY INDEX = J
   VERIFY O, Present_Value = Stages[X].Values[J]

7.3.2.X66.6 Stage Value Bitstring is Same Length as Target_References
Reason for Change: No test exists for this functionality.

Purpose: To verify that the bitstring length for the Values component of each stage is equal and corresponds to the number of entries in the Target_References property.

Test Concept: For each staging object in the IUT, the Stages and Target_References properties are read. For each object, the length of the 'Values' bitstring from the first stage is extracted. This length is compared to the length of the 'Values' bitstring in every other stage and the size of the Target_References property to verify equality.

Configuration Requirements: None

Test Steps:

```
1. REPEAT O = (each Staging object in the IUT) DO {
       READ NS = O, Stages, ARRAY INDEX = 0
       READ STG1 = Stages, ARRAY INDEX = 1
       NUMBITS = (number bits in STG1.Values)
       REPEAT N = (2 through NS) DO {
           -- check that the length of Stages[1].Values equals length of Stages[N].Values.
           READ STGN = Stages, ARRAY INDEX = N
           IF number of bits in STGN.Values <> NUMBITS THEN
               ERROR "Length of the Values bitstrings are not the same in all stages."
           }
       VERIFY Target_References = NUMBITS, ARRAY_INDEX = 0
       }
```

7.3.2.X66.7 Max_Pres_Value Equals Last Stage Limit
Reason for Change: No test exists for this functionality.

Purpose: To verify that Max_Pres_Value is equivalent to the Limit defined in the last Stage.

Test Concept: Max_Pres_Value is read and checked for equality with the Limit defined in the last element of the Stages array.

Configuration Requirements: None

Test Steps:

1. READ N = Stages, ARRAY INDEX = 0
2. VERIFY Max_Pres_Value = Stages[N].Limit

7.3.2.X66.8 CONFIGURATION_ERROR when Min_Pres_Value is too Large
Reason for Change: No test exists for this functionality.

Purpose: To verify that Reliability has a value of CONFIGURATION_ERROR when Min_Pres_Value has a value greater than Stages[1].Limit - Stages[1].Deadband.

Test Concept: Min_Pres_Value is made to exceed the value of Stages[1].Limit - Stages[1].Deadband by first writing directly to Min_Pres_Value, then by making a change to Stages[1].Limit, and then by making a change to Stages[1].Deadband. After each modification, if it is successful, Reliability is verified to have a value of CONFIGURATION_ERROR and then the modification is reversed, and Reliability is verified to have a value of NO_FAULT_DETECTED.

Configuration Requirements: At the start of the test, the Staging object used for this test, O1, shall be properly configured such that Reliability = NO_FAULT_DETECTED. At the start of the test, Present_Value shall be equal to Min_Pres_Value.

Test Steps:

1. READ MINPV1 = Min_Pres_Value
2. VERIFY Present_Value = MINPV1
3. VERIFY Reliability = NO_FAULT_DETECTED
4. READ STG1 = Stages, ARRAY INDEX = 1
5. SL = STG1.Limit
6. SV = STG1.Values
7. SD = STG1.Deadband
8. IF (Min_Pres_Value is writable) THEN
   TRANSMIT WriteProperty-Request,
   'Object Identifier' = O1
   'Property Identifier' = Min_Pres_Value ,
   'Property Value' = (MINPV2: where MINPV2 > (SL-SD))
   RECEIVE BACnet-SimpleACK-PDU |
   (BACnet-Error-PDU,
   'Error Class' = Property,
   'Error Code' = VALUE_OUT_OF_RANGE)
   IF (a BACnet-Simple-ACK-PDU was received) THEN
   VERIFY Min_Pres_Value = MINPV2
   VERIFY Reliability = CONFIGURATION_ERROR
   VERIFY Present_Value = MINPV2
   VERIFY Present_Stage = 1
   WRITE Min_Pres_Value = MINPV1
   VERIFY Reliability = NO_FAULT_DETECTED
   ELSE
   VERIFY Present_Value = MINPV1
   VERIFY Reliability = NO_FAULT_DETECTED
9. IF (Stages is writable) THEN
   TRANSMIT WriteProperty-Request,
   'Object Identifier' = (the staging object under test),
   'Property Identifier' = Stages,
   'Property Array Index' = 1,
   'Property Value' = {
   Limit = (NL: where NL-SD < Min_Pres_Value),
   Values = SV,
   DeadBand = SD
   }
   RECEIVE BACnet-SimpleACK-PDU |
   (BACnet-Error-PDU,
   'Error Class' = PROPERTY,
   'Error Code' = VALUE_OUT_OF_RANGE)
   IF (a BACnet-SimpleACK-PDU was received) THEN
   VERIFY Stages = { Limit=NL, Values=SV, Deadband=SD }, ARRAY INDEX = 1
   VERIFY Reliability = CONFIGURATION_ERROR
   VERIFY Present_Value = MINPV1
   VERIFY Present_Stage = 1
   WRITE Stages = { Limit=SL, Values=SV, Deadband=SD }, ARRAY INDEX = 1
   VERIFY Reliability = NO_FAULT_DETECTED
   ELSE
   VERIFY Stages = { Limit=SL, Values=SV, Deadband=SD }, ARRAY INDEX = 1
   VERIFY Reliability = NO_FAULT_DETECTED
   TRANSMIT WriteProperty-Request,
   'Object Identifier' = (the staging object under test),

```
        'Property Identifier' =         Stages,
        'Property Array Index' =        1,
        'Property Value' =              {
                                        Limit=SL,
                                        Values=SV,
                                        Deadband=(ND: where SL-ND < Min_Pres_Value)
                                        }
    RECEIVE BACnet-SimpleACK-PDU |
    (BACnet-Error-PDU,
        'Error Class' = PROPERTY,
        'Error Code' = VALUE_OUT_OF_RANGE)
    IF (a BACnet-SimpleACK-PDU was received) THEN
        VERIFY Stages = { Limit=SL, Values=SV, Deadband=ND }, ARRAY INDEX = 1
        VERIFY Reliability = CONFIGURATION_ERROR
        VERIFY Present_Value = MINPV1
        VERIFY Present_Stage = 1
        WRITE Stages = { Limit=SL, Values=SV, Deadband=SD }, ARRAY INDEX = 1
        VERIFY Reliability = NO_FAULT_DETECTED
    ELSE
        VERIFY Stages = { Limit=SL, Values=SV, Deadband=SD }, ARRAY INDEX = 1
        VERIFY Reliability = NO_FAULT_DETECTED
```

7.3.2.X66.9 COMMUNICATION_FAILURE on Failed Write to External Target Reference
Reason for Change: No test exists for this functionality.

Purpose: To verify that Reliability is set to COMMUNICATION_FAILURE when an attempt to write to a remote target fails.

Test Concept: The Staging object is configured with a Target_Reference aimed at an object in the TD. The Staging object's Present_Value is written such that a change to Present_Stage occurs. When the external target property is written by the IUT, the TD shall not respond. The test verifies that the write to Present_Value returns a Result(+) and Reliability is set to COMMUNICATION_FAILURE.

Configuration Requirements: The Staging object used for this test shall be configured with at least one object, O1, in the Target_References property which is located in the TD. The Stages property shall be configured with two stages such that Stages[S].Values = {…V1…} and Stages[S+1].Values = {…V2…} where V1 and V2 correspond to the target, O1, and V1 <> V2. At the start of the test, the Staging object is properly configured such that Reliability = NO_FAULT_DETECTED and Present_Stage = S. If no Staging object in the IUT supports external references in the Target_References property, this test shall be skipped.

Test Steps:

1. READ S = Present_Stage
2. WRITE Present_Value = (X: a value that will change Present_Stage to S+1)
3. RECEIVE WriteProperty-Request,
   'Object Identifier' = O1,
   'Property Identifier' = Present_Value,
   'Property Value' = V2
4. WAIT (Number_Of_APDU_Retries + 1) * (**Internal Processing Fail Time** + APDU_Timeout)
5. VERIFY Reliability = COMMUNICATION_FAILURE

7.3.2.X66.10 Fault Indicated on Failed Write to Local Target Reference
Reason for Change: No test exists for this functionality.

Purpose: To verify that Reliability is set when an attempt to write to a local target fails.

Test Concept: The Staging object is configured with a Target_Reference aimed at an object in the IUT which is not writable or non-existent. The Staging object's Present_Value is written such that a change to Present_Stage occurs. The test verifies that the write to the Staging object's Present_Value returns a Result(+) and Reliability is set to indicate a failure to write one of the targets.

Configuration Requirements: The Staging object used for this test shall be configured with at least one object, O1, in the Target_References property which is local to the IUT, yet not writable or non-existent. The Stages property shall be configured with two stages such that Stages[S].Values = {…V1…} and Stages[S+1].Values = {…V2…} where V1 and V2 correspond to the target, O1, and V1 <> V2. At the start of the test, the Staging object is properly configured such that Reliability =

NO_FAULT_DETECTED and Present_Stage = S. If no Staging object can be configured to reference a non-writable or non-existent local object, this test shall be skipped.

Test Steps:

1. READ S = Present_Stage

-- cause the staging object to write to the non-existent or non-writable target object

2. WRITE Present_Value = (X: a value that will change Present_Stage to S+1)
3. VERIFY Reliability <> NO_FAULT_DETECTED

7.3.2.X66.11 Out_Of_Service, Status_Flags, and Reliability for Staging Object

Reason for Change: No test exists for this functionality.

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

7.3.2.X66.12 Stages Array Sizing Test

Reason for Change: No test exists for this functionality.

Purpose: This test case verifies that, when the size of the of the Stages array is changed by writing to the ARRAY INDEX, the size of the array changes accordingly and any new entries are properly initialized.

Test Concept: The Stages array is increased by writing the array size. It is verified that the Stages property is extended and that the new entries contain 'Limit' = 0.0, 'Values' = {0…0}, and 'Deadband' = 0.0. Reliability is verified to be CONFIGURATION_ERROR. Present_Stage is verified to be 1. Present_Value is verified to be Min_Pres_Value. If the Stage_Names property is present, the size of the array is checked to verify that it matches the size of the Stages array.

Throughout the test, the array size of the Stage_Names property is checked to verify it is consistent with the array size of the Stages property.

Configuration Requirements: At the start of the test, the Staging object is properly configured such that Reliability = NO_FAULT_DETECTED and Present_Stage = S. The size of the Stages array is greater than 1 and less than the maximum array size.

Test Steps:

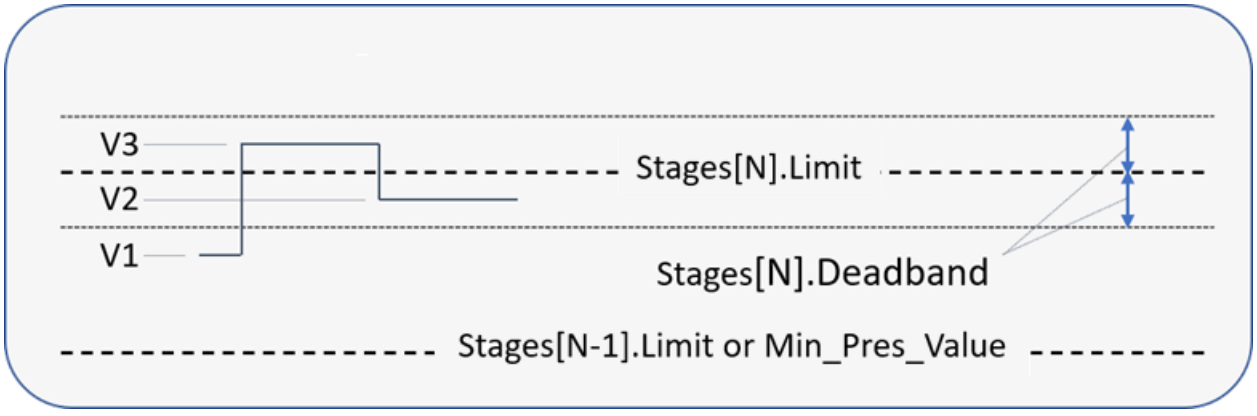
1. READ N = Stages, ARRAY INDEX = 0
2. IF (Stage_Names is present) THEN {
   VERIFY = Stage_Names = N, ARRAY INDEX = 0
   }
3. READ NT = Target_References, ARRAY INDEX = 0
4. WRITE Stages = N+X, ARRAY INDEX = 0 -- where (X ≥ 1)
5. VERIFY Stages = N+X, ARRAY INDEX = 0
6. VERIFY Stages = (0.0,{0…0},0.0), ARRAY INDEX = N+X -- where the number of bits in Values is NT
7. VERIFY Reliability = CONFIGURATION_ERROR
8. VERIFY Present_Stage = 1
9. READ MV = Min_Pres_Value
10. VERIFY Present_Value = MV
11. IF (Stage_Names is present) THEN {
    VERIFY Stage_Names = N+X, ARRAY INDEX = 0
    WRITE Stages = N, ARRAY INDEX = 0
    VERIFY Stage_Names = N, ARRAY INDEX = 0
    }

7.3.2.X66.13 Present_Stage Evaluates on Change to Stages Property

Reason for Change: No test exists for this functionality.

Purpose: To verify that Present_Stage gets re-evaluated when the Stages property is changed.

Test Concept: Present_Value is written with a value, V3, that exceeds Stages[N].limit but does not exceed the deadband threshold and cause a change to Present_Stage. The Stages property is written with a new value such that Stage[N] is unaffected by the change. Present_Stage is read to verify that it is now (N+1). Present_Value is then written with a value, V2, that is below Stages[N].limit but above the deadband threshold so Present_Stage remains at (N+1). The Stages property is written with a new value such that Stage[N] is unaffected by the change. Present_Stage is read to verify that it is now N.



Configuration Requirements: The Staging object used for this test must be configured with at least two stages. Deadband shall be configured with a non-zero value for stage N (Stage[N].Deadband <> 0). If deadband for stage N cannot be configured this way, this test shall be skipped. At the start of the test, the Staging object is configured with Present_Value = V1 and Present_Stage = N.

Test Steps:

1. VERIFY Present_Stage = N
2. VERIFY Present_Value = V1
3. READ STAGES1 = Stages
4. WRITE Present_Value = (V3: Stages[N].Limit < V3 < (Stages[N].Limit + Stages[N].Deadband))
5. VERIFY Present_Stage = N
6. IF (Stages is writable) THEN
   WRITE Stages = (STAGES2: any valid value different from STAGES1 but
   with the same value for Stage[N])
   ELSE
   MAKE Stages = (STAGES2: any valid value different from STAGES1but
   with the same value for Stage[N])
7. VERIFY Present_Value = V3
8. VERIFY Present_Stage = N+1
9. WRITE Present_Value = (V2: (Stages[N].Limit - Stages[N].Deadband) < V2 < Stages[N].Limit)
10. VERIFY Present_Stage = N+1
11. IF (Stages is writable) THEN
    WRITE Stages = (STAGES3: any valid value different from STAGES2 but
    with the same value for Stage[N])
    ELSE
    MAKE Stages = (STAGES3: any valid value different from STAGES2 but
    with the same value for Stage[N])
12. VERIFY Present_Value = V2
13. VERIFY Present_Stage = N

7.3.2.X66.14 CONFIGURATION_ERROR when Limits are Out of Order

Reason for Change: No test exists for this functionality.

Purpose: To verify that Stages defined in the staging object are arranged in ascending order and, if not, Reliability is set to CONFIGURATION_ERROR.

Test Concept: Write Stages out of order; use specific values that violate the limit value ascension rule. Verify that the object identifies the problem and sets Reliability.

Configuration Requirements: At the start of the test, the Staging object is properly configured such that Reliability = NO_FAULT_DETECTED.

Test Steps:

1. VERIFY Reliability = NO_FAULT_DETECTED
2. READ NS = Stages, ARRAY INDEX = 0
3. N = (any value where 1 <= N < NS)
4. WRITE Stages = {
   Limit = (LIM: where LIM > Stages[N+1].Limit),
   Values = (any valid value of the correct length),
   Deadband = (any valid value)
   }, ARRAY INDEX = N
5. VERIFY Reliability = CONFIGURATION_ERROR
6. VERIFY Present_Value = Min_Pres_Value
7. VERIFY Present_Stage = 1

7.3.2.X66.15 CONFIGURATION_ERROR when Deadband < 0

Reason for Change: No test exists for this functionality.

Purpose: To verify that Stages defined in the staging object do not have a Deadband value less than 0, or if they do, when Deadband is less than 0, Reliability is set to CONFIGURATION_ERROR.

Test Concept: Write an entry in the Stages property, changing the deadband to a negative value. Verify that the either the write fails, or that Reliability is set to CONFIGURATION_ERROR.

Configuration Requirements: At the start of the test, the Staging object is properly configured such that Reliability = NO_FAULT_DETECTED.

Test Steps:

1. VERIFY Reliability = NO_FAULT_DETECTED
2. READ NS = Stages, ARRAY INDEX = 0
2. N = (any value where 1 <= N < NS)
3. TRANSMIT WriteProperty-Request,
    'Object Identifier' = (the staging object under test),
    'Property Identifier' = Stages,
    'Property Array Index' = N,
    'Property Value' = {
        Limit = Stages[N].Limit,
        Values = Stages[N].Values,
        Deadband = (any negative value)
        }
4. RECEVE BACnet-SimpleACK-PDU |
        (BACnet-Error-PDU,
            'Error Class' = PROPERTY,
            'Error Code' = VALUE_OUT_OF_RANGE)
5. IF (a BACnet-SimpleACK-PDU was received) THEN {
    VERIFY Reliability = CONFIGURATION_ERROR
    VERIFY Present_Value = Min_Pres_Value
    VERIFY Present_Stage = 1
}

7.3.2.X66.16 CONFIGURATION_ERROR when Stages Size is less than Two
Reason for Change: No test exists for this functionality.

Purpose: To verify that the Stages array has a minimum length of two, and if not, Reliability is set to CONFIGURATION_ERROR.

Test Concept: Write the Stages property, without an array index, setting the length of the array to 1. Verify that the either the write fails, or that Reliability is set to CONFIGURATION_ERROR.

Configuration Requirements: At the start of the test, the Staging object is properly configured such that Reliability = NO_FAULT_DETECTED.

Test Steps:

1. VERIFY Reliability = NO_FAULT_DETECTED
2. TRANSMIT WriteProperty-Request,
    'Object Identifier' = (the staging object under test),
    'Property Identifier' = Stages,
    'Property Array Index' = 0,
    'Property Value' = (0 or 1)
3. RECEVE BACnet-SimpleACK-PDU |
        (BACnet-Error-PDU,
            'Error Class' = PROPERTY,
            'Error Code' = VALUE_OUT_OF_RANGE)
5. IF (a BACnet-SimpleACK-PDU was received) THEN {
    VERIFY Reliability = CONFIGURATION_ERROR
    VERIFY Present_Value = Min_Pres_Value
    VERIFY Present_Stage = 1
}

7.3.2.X66.17 Stage_Names and Stages Size Equality Test
Reason for Change: No test exists for this functionality.

Purpose: To verify that the size of the Stage_Names array is equal to the size of the Stages array.

Test Concept: Verify that the Stages array and Stage_Names array are of the same length.

Test Steps:

1. READ N = Stages, ARRAY INDEX = 0
2. VERIFY Stage_Names = N, ARRAY INDEX = 0

7.3.2.X66.18 Stage_Names Array Sizing Test
Reason for Change: No test exists for this functionality.

Purpose: This test case verifies that, when the size of the Stage_Names array is changed by writing to the ARRAY INDEX, the size of the array and the size of the Stages array changes accordingly and new values added to the Stages array are initialized to contain 'Limit' = 0.0, 'Values' = {0…0}, and 'Deadband' = 0.0, and Reliability is set to CONFIGURATION_ERROR.

Test Concept: Resize the Stage_Names array larger by writing the size and verify that Stages is also resized. Shrink the array back and verify Stages. Resize once more by writing the whole array and verify that Stages resizes correctly. Each time verify that new stages are correct.

Configuration Requirements: If the Stages property is not resizable by writing to it, this test shall be skipped.

Test Steps:

1. READ N = Stage_Names, ARRAY INDEX = 0
2. WRITE Stage_Names = N+1, ARRAY INDEX = 0
3. VERIFY Stages = N+1, ARRAY INDEX = 0
4. VERIFY Stages = {Limit=0.0, Values={0…0}, Deadband=0.0}, ARRAY INDEX = N+1
5. WRITE Stage_Names = N, ARRAY INDEX = 0
6. VERIFY Stages = N, ARRAY INDEX = 0

7. WRITE Stage_Names = (an array, of strings, with a length, N2, which the IUT will accept other than N)
8. VERIFY Stages = N2, ARRAY INDEX = 0
9. VERIFY Stages = (an array of length N2 of stages consistent with the object's configuration)
10. IF (N2 > N) THEN {
    REPEAT J = (N … N2) {
        VERIFY Stages = {Limit=0.0, Values={0…0}, Deadband=0.0}, ARRAY INDEX = J
    }
}

7.3.2.X66.19 Target References Array Sizing Test
Reason for Change: No test exists for this functionality.

Purpose: To verify that a change to size of the Target_References array results in an equivalent change to the length of the 'Values' portion of all elements of the Stages property and that new bits in the 'Values' are set to '0'.

Test Concept: Resize the Target_References array larger, and verify that the values field in each stage is updated with new bits. Resize the array smaller and verify that the values field in each stage is resized smaller.

Configuration Requirements: The staging object is configured with at least 1 Target_Reference.

Test Steps:

1. READ STAGES1 = Stages
2. NTR = (length of STAGES1[0].Values)
3. WRITE Target_References = NTR+1, ARRAY INDEX = 0
4. REPEAT J = (1 … STAGES1[0]) DO {
    VERIFY Stages = {
        Limit=STAGES1[J].Limit,
        Values=(the value of STAGES1[J].Values with 1 more 0 tacked on the end),
        DeadBand=STAGES1 [J].Deadband
    }, ARRAY INDEX = J
}

5. WRITE Target_References = NTR, ARRAY INDEX = 0
6. REPEAT J = (1 … STAGES1[0]) DO {
    VERIFY Stages = {
        Limit=STAGES1[J].Limit,

Values=(the value of STAGES1[J].Values),
DeadBand=STAGES1[J].Deadband
}, ARRAY INDEX = J
}

7.3.2.X66.20 Writing Target_References with an Unsupported External Reference
Reason for Change: No test exists for this functionality.

Purpose: To verify the correct Result(-) when Target_References does not support objects in an external device.

Test Concept: Attempt writing Target_References of a Staging object with an external object reference. Verify the IUT returns the correct Result(-).

Configuration Requirements: The IUT is configured with a Staging Value object which does not support references to external objects. If the IUT cannot be configured this way, this test shall be skipped.

Test Steps:

1. READ X = Target_References
2. TRANSMIT WriteProperty-Request,
   'Object Identifier' = (the staging object under test),
   'Property Identifier' = Target_References,
   'Property Array Index' = 1,
   'Property Value' = (a reference to a binary object in the TD)

3. RECEIVE BACnet-Error-PDU,
   'Error Class' = PROPERTY
   'Error Code' = OPTIONAL_FUNCTIONALITY_NOT_SUPPORTED

[Move test 9.2.1.1 from 135.1 into BTL Specified Tests and modify]
9.2.1.1 Change of Value Notifications
Reason for Change: The existing test did not account for other properties which are expected for certain object types.

Purpose: To verify that the IUT can execute ConfirmedCOVNotification requests from object types that provide the Present_Value and Status_Flags properties in COV notifications.

Test Concept: The IUT is made to subscribes for COV from an object of the type being tested. The TD then sends a COV notification to the IUT and verifies that the IUT exhibits any actions identified by the vendor.

Test Steps:

1. RECEIVE SubscribeCOV,
   'Subscriber Process Identifier' = (any valid process identifier),
   'Monitored Object Identifier' = X,
   'Issue Confirmed Notifications ' = TRUE,
   'Lifetime' = (a value greater than one minute)
2. TRANSMIT BACnet-SimpleACK-PDU
3. TRANSMIT ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = (the process identifier used in step 2),
   'Initiating Device Identifier' = TD,
   'Monitored Object Identifier' = X,
   'Time Remaining' = (the time remaining in the subscription),
   'List of Values' = (Present_Value ~~and~~ , Status_Flags*, and additional properties* appropriate to object type X)
4. RECEIVE BACnet-SimpleACK-PDU
5. CHECK (to ensure that any appropriate functions defined by the manufacturer, such as displaying information on a workstation screen are carried out)

[Move test 9.3.2 from 135.1 into BTL Specified Tests and modify]

9.3.2 Change of Value Notifications
Reason for Change: The existing test did not account for other properties which are expected for certain object types.

Purpose: To verify that the IUT can execute UnconfirmedCOVNotification requests from objects that provide the Present_Value and Status_Flags properties in COV notifications.

Test Concept: The IUT is made to subscribes for COV from an object of the type being tested. The TD then sends a COV notification to the IUT and verifies that the IUT exhibits any actions identified by the vendor.

Test Steps:

1. RECEIVE SubscribeCOV,
   'Subscriber Process Identifier' = (any valid process identifier),
   'Monitored Object Identifier' = X,
   'Issue Confirmed Notifications ' = FALSE,
   'Lifetime' = (a value greater than 1 minute)
2. TRANSMIT BACnet-SimpleACK-PDU
3. TRANSMIT UnconfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = (the process identifier used in step 2),
   'Initiating Device Identifier' = TD,
   'Monitored Object Identifier' = X,
   'Time Remaining' = (the time remaining for the subscription),
   'List of Values' = (Present_Value~~, and~~ Status_Flags*, and additional properties* appropriate to object type X)
4. CHECK (to ensure that any appropriate functions defined by the manufacturer, such as displaying information on a workstation screen are carried out)

[Add new 8.2.* and 8.3.* tests for Staging Objects into BTL Specified Tests]

8.2.X17 Change of Value Notification of Staging Object Present_Value Property
Reason for Change: No test exist for this functionality.

Purpose: To verify that the IUT can initiate ConfirmedCOVNotification service requests conveying a change of the Present_Value property in Staging objects that support COV_Increment.

Test Concept: A CPV subscription 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.

Configuration Requirements: Select a Staging object where Present_Value is not expected to change outside the tester's control. The object is configured such that the change in Present_Value required to change stages is larger than COV_Increment. If the IUT cannot be configured with such a Staging object, this test shall be skipped.

Test Steps:

1. READ PV1 = Present_Value
2. READ SF1 = Status_Flags
3. READ PS1 = Present_Stage

-- subscribe for COV and receive initial notification
4. TRANSMIT SubscribeCOV-Request,
   'Subscriber Process Identifier' = (PID1: any value > 0 chosen by the TD),
   'Monitored Object Identifier' = X,
   'Issue Confirmed Notifications' = TRUE,
   'Lifetime' = L
5. RECEIVE BACnet-SimpleACK-PDU
6. BEFORE **Notification Fail Time**
   RECEIVE ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = PID1,
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = X,
   'Time Remaining' = (any value appropriate for the Lifetime selected),
   'List of Values' = (PV1, SF1, PS1)
7. TRANSMIT BACnet-SimpleACK-PDU

-- change Present_Value by less than COV_Increment, and not enough to change the stage
8. WRITE X, Present_Value = (PV2: a value that differs from PV1 by less than COV_Increment and which is in the range for the current stage)
9. WAIT **Notification Fail Time**
10. CHECK (verify that no COV notification was transmitted)

-- change Present_Value by more than COV_Increment, but not enough to change the stage
11. WRITE X, Present_Value = (PV3: a value that differs from PV1 by an amount greater than COV_Increment and which is in the range for the current stage)
12. BEFORE **NotificationFailTime**
    RECEIVE ConfirmedCOVNotification-Request,
        'Subscriber Process Identifier' = PID1,
        'Initiating Device Identifier' = IUT,
        'Monitored Object Identifier' = X,
        'Time Remaining' = (any value appropriate for the Lifetime selected),
        'List of Values' = (PV3, SF1, PS1)
13. TRANSMIT BACnet-SimpleACK-PDU

-- cleanup the subscription
14. TRANSMIT SubscribeCOV-Request,
        'Subscriber Process Identifier' = PID1,
        'Monitored Object Identifier' = X
15. RECEIVE BACnet-SimpleACK-PDU

8.2.X18 Change of Value Notification from a Staging Object Status_Flags Property
Reason for Change: No test existing or this functionality.

Purpose: To verify that the IUT can initiate ConfirmedCOVNotification service requests conveying a change of the Status_Flags property of Staging objects.

Test Concept: A COV subscription 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 Status_Flags property of the monitored object is then changed and a notification shall be received. The value of the Status_Flags property can be changed by using the WriteProperty service or by another means. For implementations where it is not possible to write Out_Of_Service or change the Status_Flags by any other means, this test shall be skipped.

Configuration Requirements: Select a Staging object where Present_Value is not expected to change outside the tester's control.

Test Steps:

1. VERIFY Out_Of_Service = FALSE
2. READ PV1 = Present_Value
3. READ SF1 = Status_Flags
4. READ PS1 = Present_Stage

-- subscribe for COV and receive initial notification
5. TRANSMIT SubscribeCOV-Request,
        'Subscriber Process Identifier' = (PID1: any value > 0 chosen by the TD),
        'Monitored Object Identifier' = X,
        'Issue Confirmed Notifications' = TRUE,
        'Lifetime' = L
6. RECEIVE BACnet-SimpleACK-PDU
7. BEFORE Notification Fail Time
    RECEIVE ConfirmedCOVNotification-Request,
        'Subscriber Process Identifier' = PID1,
        'Initiating Device Identifier' = IUT,
        'Monitored Object Identifier' = X,
        'Time Remaining' = (any value appropriate for the Lifetime selected),
        'List of Values' = (PV1, SF1, PS1)
8. TRANSMIT BACnet-SimpleACK-PDU

-- change Status_Flags and receive notification
9. IF Out_Of_Service is writable THEN
   WRITE X, Out_Of_Service = TRUE
   SF2 = (SF1 with the Out_Of_Service bit changed to 1)
   ELSE
   MAKE (Status_Flags = SF2, any value other than SF1)
10. BEFORE Notification Fail Time
   RECEIVE ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = PID1,
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = X,
   'Time Remaining' = (any value appropriate for the Lifetime selected),
   'List of Values' = (PV1, SF2, PS1)
11. TRANSMIT BACnet-SimpleACK-PDU

-- cleanup the subscription and Out_Of_Service
12. TRANSMIT SubscribeCOV-Request,
   'Subscriber Process Identifier' = PID1,
   'Monitored Object Identifier' = X
13. RECEIVE BACnet-SimpleACK-PDU
14. IF (Out_Of_Service was changed via writing) THEN
   WRITE X, Out_Of_Service = FALSE

8.2.X19 Change of Value Notification from a Staging Object Present_Stage Property
Reason for Change: No test existing or this functionality.

Purpose: To verify that the IUT can initiate ConfirmedCOVNotification service requests conveying a change of the Present_Stage property of Staging objects.

Test Concept: A COV subscription 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_Stage property of the monitored object is then changed and a notification shall be received.

Configuration Requirements: Select a Staging object, O1, where Present_Value is not expected to change outside the tester's control. The object shall be configured with Present_Value having a value, PV1, which is less then COV_Increment away from a value, PV2, which will change the current stage to a new stage, PS2. If no Staging object can be configured with a COV_increment larger than the resolution of Present_Value, this test shall be skipped.

Test Steps:

1. VERIFY Present_Value = PV1
2. VERIFY Present_Stage = PS2
3. READ SF1 = Status_Flags
4. CHECK(The difference between PV1 and PV2 is less than COV_Increment)

-- subscribe for COV and receive initial notification
5. TRANSMIT SubscribeCOV-Request,
   'Subscriber Process Identifier' = (PID1: any value > 0 chosen by the TD),
   'Monitored Object Identifier' = O1,
   'Issue Confirmed Notifications' = TRUE,
   'Lifetime' = L
6. RECEIVE BACnet-SimpleACK-PDU
7. BEFORE Notification Fail Time
   RECEIVE ConfirmedCOVNotification-Request,
   'Subscriber Process Identifier' = PID1,
   'Initiating Device Identifier' = IUT,
   'Monitored Object Identifier' = O1,
   'Time Remaining' = (any value appropriate for the Lifetime selected),
   'List of Values' = (PV1, SF1, PS1)
8. TRANSMIT BACnet-SimpleACK-PDU

-- change Present_Value and receive notification
9. WRITE X, Present_Value = PV2

10. BEFORE Notification Fail Time
    RECEIVE ConfirmedCOVNotification-Request,
        'Subscriber Process Identifier' = PID1,
        'Initiating Device Identifier' = IUT,
        'Monitored Object Identifier' = O1,
        'Time Remaining' = (any value appropriate for the Lifetime selected),
        'List of Values' = (PV2, SF1, PS2)
11. TRANSMIT BACnet-SimpleACK-PDU

-- cleanup the subscription
12. TRANSMIT SubscribeCOV-Request,
        'Subscriber Process Identifier' = PID1,
        'Monitored Object Identifier' = O1
13. RECEIVE BACnet-SimpleACK-PDU

8.3.X17 Change of Value Notification of Staging Object Present_Value Property
Reason for Change: No test exists for this functionality

Purpose: To verify that the IUT can initiate UnconfirmedCOVNotification service requests conveying a change of the Present_Value property of Staging objects.

Test Steps: The steps for this test case are identical to the test steps in 8.2.X17 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.

8.3.X18 Change of Value Notification of Staging Object Status_Flags Property
Reason for Change: No test exists for this functionality

Purpose: To verify that the IUT can initiate UnconfirmedCOVNotification service requests conveying a change of the Status_Flags property of Staging objects.

Test Steps: The steps for this test case are identical to the test steps in 8.2.X18 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.

8.3.X19 Change of Value Notification of Staging Object Present_Stage Property
Reason for Change: No test exists for this functionality

Purpose: To verify that the IUT can initiate UnconfirmedCOVNotification service requests conveying a change of the Present_Stage property of Staging objects.

Test Steps: The steps for this test case are identical to the test steps in 8.2.X18 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.