NISTIR 4563

A Draft Abstract Test Suite for Determining Conformance to the BACnet Protocol

U.S. Department of Commerce
National Institute of Standards and Technology
Building and Fire Rearch Laboratory
Gaithersburg, MD 20899

NISTIR 4563

A Draft Abstract Test Suite for Determining Conformance to the BACnet Protocol

Steve T. Bushby

April 1991

U.S. Department of Commerce
Robert A. Mosbacher, Secretary
National Institute of Standards and Technology
John W. Lyons, Director
Building and Fire Rearch Laboratory
Gaithersburg, MD 20899

ABSTRACT

The BACnet communication protocol for building automation and control systems is in an advanced state of development and is expected to be released for public review in early 1991. When the review process is completed it will become an ASHRAE standard. One of the important outstanding issues to be resolved is conformance to the standard and how to test devices to determine if they meet the conformance requirements. This report is a draft Abstract Test Suite based on working draft 3 of the proposed standard.

This Abstract Test Suite is a first step in developing the tests which will be used to certify conformance to BACnet. Its purpose is to provide a starting point from which a conformance certification program can be built and to focus discussions on the outstanding conformance issues that need to be resolved before the standard can be considered complete. The role of an abstract test suite in the conformance testing process is described, a proposed BACnet test system architecture is presented and individual abstract test cases are defined. Test cases to determine support for object types and application services defined in working draft 3 [9] are included. A standard object configuration is also proposed to simplify the testing process.

CONTENTS

1. INTRODUCTION

Since January, 1987 the American Society of Heating, Refrigerating, and Air-Conditioning Engineers (ASHRAE) has been actively developing an industry consensus standard communication protocol for building automation and control systems. The title of the proposed standard is "BACnet - A Data Communication Protocol for Building Automation and Control Networks." BACnet is expected to be issued in draft form for public review and comment in early 1991. General information about BACnet has been published in the ASHRAE literature [1,2] and in other building trade publications [3].

The National Institute of Standards and Technology (NIST) has been actively involved in the development of BACnet from its beginning in 1987. One role that NIST has taken on is to develop procedures for testing a building automation device for conformance to BACnet. It is expected that these procedures will become the basis for an industry certification program. An architecture for the test system, based on a variation of the "coordinated abstract test method" [4], has been proposed . This test method is one of several being proposed as an international standard for testing conformance to communication protocols [5]. A methodology for developing the actual test suite has also been proposed [6].

This document presents a draft abstract test suite for testing conformance to BACnet. It is based on working draft 3 of the BACnet protocol and the modifications to working draft 3 contained in [7]. Working draft 3 was prepared by ASHRAE stndards procect committee 135P (SPC 135P) for internal use and is not a complete draft standard. Since it has not been approved for public review and comment, there may be some important changes and additions to BACnet before it is released for review. In fact, one of the issues that has not been completely resolved by the SPC is what portions of the standard must be supported in order to claim conformance to BACnet and how many different classes of conformance there will be. This draft abstract test suite contains tests for static conformance requirements for all object types defined in working draft 3 and a portion of the dynamic conformance requirenents for the defined application services.

These uncertainties present obvious difficulties to anyone attempting to develop conformance tests for BACnet at this time. Nevertheless, there are important advantages to defining conformance tests for BACnet in its present form. The process of designing the tests and building an implementation provides important feedback to the SPC about ambiguities in the draft standard, helps to bring into focus outstanding issues regarding conformance requirements, and makes it more likely that an accepted abstract test suite and conformance testing procedure will be ready when the first commercial products using BACnet become available. The abstract test suite described in this report will certainly undergo several changes as the standard approaches final form.

Section 2 of this report briefly describes what an abstract test suite is and the role it plays in developing a conformance testing implementation. Section 3 describes the test system architecture that has been assumed for the purpose of developing these abstract tests. A standard object database which plays an important role in interpreting the test results is described in section 4. Section 5 describes the conventions used to name the abstract test cases. The abstract test cases are found in sections 6 and 7. Section 6 contains the capability tests which are used to determine compliance with the static conformance requirements of the standard. For BACnet this amounts to testing the support of particular standard object-types. Section 7 contains the behavior tests which determine compliance with the dynamic conformance requirements of the standard.

2. THE NATURE OF AN ABSTRACT TEST SUITE

An abstract test suite is an organized collection of abstract test cases. Each abstract test case is a complete and independent specification of the actions required to achieve a specific test purpose. The test cases are abstract because they are specified in terms of the service primitives defined in the protocol and they are independent of any particular hardware or software used in a real test implementation. The abstract test suite becomes the basis for a real test implementation that actually generates protocol messages and evaluates the responses received.

Each test case has a specific test purpose which corresponds to one conformance requirement of the standard. It specifies what actions should be taken by the tester, what actions are expected by the implementation under test (IUT) and how to evaluate the possible results. The test case must clearly indicate under what circumstances the test is considered to have passed, failed, or was inconclusive. Unfortunately, there are many possible outcomes that are inconclusive, and the results must be interpreted in the context of the results of other tests before a judgement can be made.

Consider a test case designed to test support for a particular object-type A. One part of this test would be to read the properties of an instance of object-type A using a ReadProperty protocol service. If an unexpected response is received from this request it could be the case that the object-type is not correctly supported. It could also be the case that the ReadProperty service is not correctly implemented. Only in the context of other attempts to use the ReadProperty service which were successful can it be determined that the problem is indeed with the support for object-type A.

This example illustrates the importance of the context of a particular test case. Many possible inconclusive results can be eliminated by making assumptions about which other test cases the implementation may have already passed. These assumptions must be clearly stated and test cases need to be executed in an order that can take advantage of this simplification.

In spite of these precautions there may be circumstances where the results are ambiguous. The judgement of a person with expertise in the protocol would be required to resolve the issue. Hopefully, most of these cases can be anticipated from the experience of developing the test suite and a real test implementation. This will allow clear guidelines for interpretation of results to be established. It is expected that ASHRAE will form a standing standards project committee (SSPC), made up of members of the committee that is drafting the standard, to resolve these kinds of issues.

3. THE BACnet TEST SYSTEM ARCHITECTURE

The BACnet protocol is based on a collapsed form of the open systems interconnection (OSI) Basic Reference Model [8]. The physical, data link, and application layers of the model are included in the current draft of BACnet. It is expected that the physical and data link layers will be implemented in hardware and will be tested by means outside of the scope of this report. This leaves only the application layer to be tested.

There is an inherent asymmetry in application layer protocols. Issuing a request for a remote device is a very different process from responding to a request that has been received. In order to test the correct implementation of an application service there must be a way to test an IUT in both roles. The logical entity which allows the tester to instruct the IUT to take particular actions is called the "upper tester" in draft international standards for conformance testing [5]. The logical entity which interacts with the IUT over the communication medium (the lower layer boundary) is called the "lower tester".

The proposed BACnet test architecture is shown in Figure 2.1. It is a variation of the "coordinated test method". In this method the activities of the upper and lower testers are coordinated by passing synchronization messages. The BACnet protocol has some built-in features which provide the functionality of the upper tester and facilitate this approach.

The upper tester consists of the instantiation of a standardized object-type, ConformTest, and an application service, called the ConformTest Service. The properties of the ConformTest object, shown in Figure 2.2, represent the information needed to construct a particular application service request. The lower tester writes to the properties of this object using the normal protocol WriteProperty service. The ConformTest application service is a command to read the properties of the ConformTest object and issue the application service request described by its properties. This mechanism provides a way for the lower tester to control application service requests issued by the IUT.

The details of the lower tester are not important from the standpoint of the abstract test suite and may be found elsewhere [4]. The ConformTest object and ConformTest application service are important to the abstract test suite

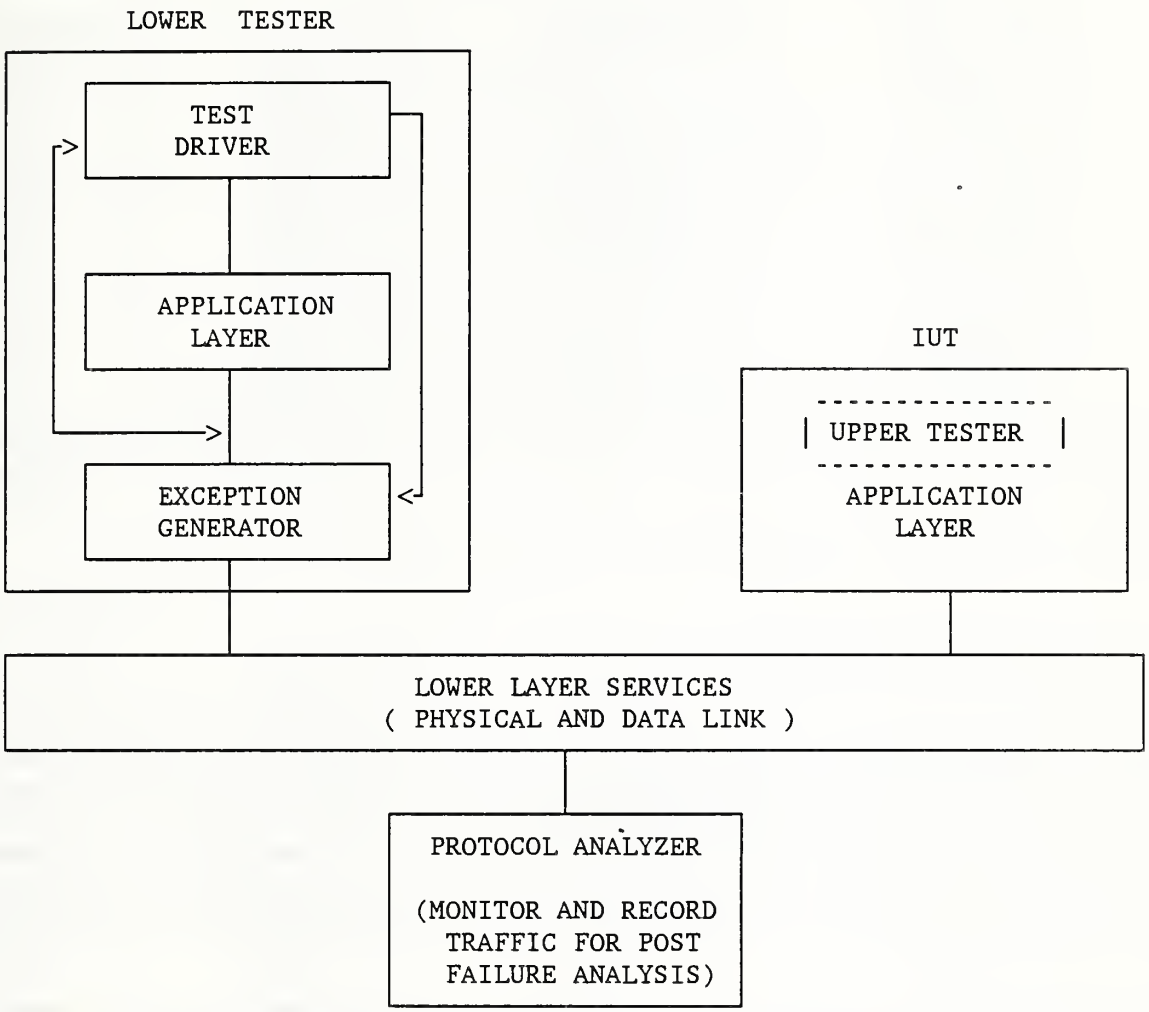

Figure 2.1 Proposed ASHRAE Abstract Conformance Test Architecture.

Property Identifier	Description
ObjectIdentifier	The name of this object i.e., "Test"
Object_Type	CONFORMTEST
Service_Name	The application service to be invoked by the IUT
Parameter1	1st parameter needed to construct APDU
Parameter2	2nd parameter needed to construct APDU
Parameter3	3rd parameter needed to construct APDU
Parameter4	4th parameter needed to construct APDU
Parameter5	5th parameter needed to construct APDU
Parameter6	6th parameter needed to construct APDU
Parameter7	7th parameter needed to construct APDU
Parameter8	8th parameter needed to construct APDU
Parameter9	9th parameter needed to construct APDU
Parameter10	10th parameter needed to construct APDU

Figure 2.2 Properties of a ConformTest Object. The Parameters represent the service specific parameters needed to construct the application protocol data unit (APDU). No BACnet service requires more than ten such parameters.

because they are the key to all of the test cases which involve application services initiated by the IUT.

4. STANDARD OBJECT CONFIGURATION

Interpreting the results of individual test cases requires knowledge of the object database contained in the device being tested. It may be necessary for the SPC to define a "standard object configuration". The standard object configuration would consist of a list of most object types defined in the protocol and a particular standard value for each property of the object. Some object types need not be included because protocol services can be used to create them. When a manufacturer submits a device for conformance testing, a protocol implementation conformance statement (PICS) would specify which object-types are supported in this device. The device would be configured such that there would be one instance of each supported object-type with the properties initialized as defined in the standard object configuration.

Without this standard object configuration it would be necessary to customize the interpretation of test results for each device tested, based on some other initial configuration supplied by the manufacturer. This approach would require a machine-readable configuration file, in a standard format, which the lower tester would process. The lower tester would, in effect, configure itself to match the configuration supplied by the manufacturer. The lower tester's database would become the reference for the "correct" property values.

In this paper it will be assumed that the standard object configuration approach is used in order to illustrate how "pass" results are interpreted. The standard object configuration is based, where practical, on the examples in the draft standard. The abstract test cases in this paper assume that all object-types defined in the standard are supported and that the IUT is configured with one instance of each of the following object-types:

Analog Input	Calendar	Event Enrollment	Schedule
Analog Output	Command	Group	
Analog Value	ConformTest	Loop	
Binary Input	Device	Multi-State Input	
Binary Output	Device Table	Multi-State Output	
Binary Value	Directory	Program	

The description which follows is a summary of the properties of these objects and their initial values.

Analog Input[1]

Key Property:	Object_Identifier	= "1AH1MAT"
Property:	Object_Type	= ANALOG_INPUT
Default Property:	Present_Value	= 58.1
Property:	Description	= "Mixed Air Temperature"
Property:	Device_Type	= "100 OHM RTD"
Property:	Status_Flags	= {FALSE, TRUE, FALSE, FALSE}
Property:	State	= NORMAL
Property:	Reliability	= RELIABLE
Property:	Out_of_Service	= FALSE
Property:	Update_Interval	= 10
Property:	Units	= "DEGF"
Property:	Min_Pres	= -50
Property:	Max_Pres	= 250
Property:	Min_Raw	= 0
Property:	Max_Raw	= 4095
Property:	Resolution	= 0.1

Analog Output

Key Property:	Object_Identifier	= "1AH1DMPR"
Property:	Object_Type	= ANALOG_OUTPUT
Default Property:	Present_Value	= 75.0
Property:	Description	= "Damper Actuator"
Property:	Device_Type	= "3-8 PSI Actuator"
Property:	Status_Flags	= {FALSE, TRUE, FALSE, FALSE}
Property:	State	= NORMAL
Property:	Reliability	= RELIABLE
Property:	Out_of_Service	= FALSE
Property:	Update_Interval	= 10
Property:	Units	= "%"
Property:	Min_Pres	= 0
Property:	Max_Pres	= 100
Property:	Min_Raw	= 0
Property:	Max_Raw	= 4095
Property:	Resolution	= 0.1

Analog Value

Key Property:	Object_Identifier	= "MA_SetPoint"
Property:	Object_Type	= ANALOG_VALUE
Default Property:	Present_Value	= 58.0

---

[1]The Present_Value of objects associated with inputs may vary because they are linked to hardware and software/firmware designed to read sensors.

Property:	Description	= "Loop Set Point"
Property:	Status_Flags	= {FALSE, TRUE, FALSE, FALSE}
Property:	State	= NORMAL
Property:	Reliability	= RELIABLE
Property:	Out_of_Service	= FALSE
Property:	Units	= "DEGF"

Binary Input[1]

Key Property:	Object_Identifier	= "HighPressSwitch"
Property:	Object_Type	= BINARY_INPUT
Property:	Description	= "Penthouse Supply High Static"
Default Property:	Present_Value	= ACTIVE
Property:	Polarity	= NORMAL
Property:	Inactive_Text	= "Static Pressure OK"
Property:	Active_Text	= "High Pressure Alarm"
Property:	Status_Flags	= {FALSE,TRUE,FALSE,FALSE}
Property:	State	= NORMAL
Property:	Reliability	= RELIABLE
Property:	Out_Of_Service	= FALSE
Property:	Change_Of_State_Time	= (90/03/23, 6, 19:01:34)
Property:	Elapsed_Active_Time	= 401
Property:	Change_Of_State_Count	= 134
Property:	Time_Of_Reset	= (90/01/01, 2, 00:00:00)

Binary Output

Key Property:	Object_Identifier	= "Floor3ExhaustFan"
Property:	Object_Type	= BINARY_OUTPUT
Property:	Description	= "Third floor bathroom exhaust fan"
Default Property:	Present_Value	= INACTIVE
Property:	Polarity	= REVERSE
Property:	Inactive_Text	= "Fan is turned off"
Property:	Active_Text	= "Fan is running"
Property:	Status_Flags	= {FALSE,TRUE,FALSE,FALSE}
Property:	State	= NORMAL
Property:	Reliability	= RELIABLE
Property:	Out_Of_Service	= FALSE
Property:	Change_Of_State_Time	= (90/03/23, 6, 19:01:34)
Property:	Run_Time	= 401
Property:	Change_Of_State_Count	= 134
Property:	Time_Of_Reset	= (90/01/01, 2, 00:00:00)
Property:	Minimum_Off_Time	= 100
Property:	Minimum_On_Time	= 10

[1]The Present_Value of objects associated with inputs may vary because they are linked to hardware and software/firmware designed to read sensors.

Binary Value

Key Property:	Object_Identifier	= "ExhaustFanEnable"
Property:	Object_Type	= BINARY_VALUE
Property:	Description	= "Exhaust Fan Operator Enable"
Default Property:	Present_Value	= ACTIVE
Property:	Inactive_Text	= "Enabled by Operator"
Property:	Active_Text	= "Fan Not Enabled by Operator"
Property:	Status_Flags	= {FALSE,TRUE,FALSE,FALSE}
Property:	State	= NORMAL
Property:	Reliability	= RELIABLE
Property:	Out_Of_Service	= FALSE
Property:	Change_Of_State_Time	= (90/03/23, 6, 19:01:34)
Property:	Elapsed_Active_Time	= 401
Property:	Change_Of_State_Count	= 134
Property:	Time_Of_Reset	= (90/01/01, 2, 00:00:00)
Property:	Minimum_Off_Time	= 0
Property:	Minimum_On_Time	= 0

Calendar

Key Property:	Object_Identifier	= "HOLIDAYS"
Property:	Object_Type	= CALENDAR
Default Property:	Present_Value	= ACTIVE
Property:	DateList	= ((90/2/19), (90/5/28), ((90/12/24), (90/1/4)))

Command

Key Property:	Object_Identifier	= "CommandTest"
Property:	Object_Type	= COMMAND
Property:	Description	= "Command Object used for Test Purposes"
Default Property	Present_Value	= 0
Property:	State0	= ((NULL,"1AH1MAT",Min_Raw,100), (NULL,"1AH1DMPR",Units,"Percent"), (NULL, "1AH1DMPR", Present_Value, 61))
Property:	State1	= ((NULL,"1AH1MAT",Min_Raw,0), (NULL,"1AH1DMPR",Units,"%"), (NULL, "1AH1DMPR", Present_Value, 75))
Property:	State2	= (NULL, "Floor3ExhaustFan", Present_Value, ACTIVE)
Property:	State3	= (NULL, "Floor3ExhaustFan", Present_Value, INACTIVE)
Property:	State4	= ((NULL,"1AH1DMPR",NULL,20), (NULL,"Floor3ExhaustFan",NULL, ACTIVE), (NULL, "MA_SetPoint", NULL, 76))
Property:	State5	= ((NULL,"1AH1DMPR",NULL,75), (NULL,"Floor3ExhaustFan",NULL, INACTIVE), (NULL, "MA_SetPoint", NULL, 58))
Property:	State6	= ((lower tester, "TestLoop", Process_Variable_Reference, ("MA_SetPoint", Present_Value)), (lower tester, "Rm208Sched", Sunday_Schedule, (0:00, INACTIVE), (8:00, ACTIVE), (2:00, INACTIVE)), (lower tester, "Fan1_Output", NULL, Sate3))
Property:	State7	= ((lower tester, "TestLoop", Process_Variable_Reference, ("1AH1MAT", Present_Value)), (lower tester, "Rm208Sched", Sunday_Schedule, (0:00,

INACTIVE)), (lower tester, "Fan1_Output", NULL, Sate2))

Device

There is no standard configuration for this object because the value of most of its properties are vendor specific. The initial configuration must be supplied by the vendor. The Device Object shall be configured as the "default" object.

ConformTest

All properties of this object shall be initialized to NULL except for the Object_Identifier property which shall be initialized to "ConformTest".

Device Table

Key Property:	Object_Identifier	= "Alarm_Consoles"
Property:	Object_Type	= DEVICE_TABLE
Property:	Description	= "Building ABC Host Computer Systems"
Property:	Device_List	= "Node 5", "Node 7"

Directory

Key Property:	Object_Identifier	= "Directory Object 79"
Property:	Object_Type	= DIRECTORY
Property:	Description	= "Outside Air Temp Obj Dir"
Property:	Reference_ObjectIdentifier	= "OA TEMP-NORTH"
Property:	Reference_ObjectType	= ANALOG_INPUT
Property:	Reference_Nickname	= "OA-MAIN"
Property:	Device_Address	= A5
Property:	Site_ID	= "North_Shore"
Property:	Tel_Number	= "312-123-4567"

Event Enrollment

Key Property:	Object_Identifier	= "Zone1ALARM"
Property:	Object_Type	= EVENT_ENROLLMENT
Property:	Event_Type	= OUT_OF_RANGE_ALARM
Property:	Property_Reference	= ("Zone1TEMP", Present_Value)
Property:	State	= ALARM
Property:	Enabled	= TRUE
Property:	Acknowledged	= FALSE
Property:	Notification_Rules	= ((NORMAL_TO_ALARM, 2, 3, TRUE, "Alarm"), (ALARM_TO_NORMAL, 5, 3, FALSE, "Normal"))
Property:	Parameter_List	= (65., 85., .25)
Property:	Confirmed_Recipient_List	= ('Node 29','Node 96',"Alarm Consoles")
Property:	Unconfirmed_Recipient_List	= (NULL)

Group

Key Property:	Object_Identifier	= "Test_Group"
Property:	Object_Type	= GROUP
Property:	List_of_Access_Specifications	= (("1AH1MAT",(Min_Pres, Max_Pres, Description)), ("MA_SetPoint",(Present_Value, Units)), ("Alarm_Consoles",(Description, Device_List)))
Property:	Present_Value	= (-50, 250, "Mixed Air Temperature", 58.0, "DEGF", "Building ABC Host Computer Systems", 'Node5', 'Node7')

Loop

Key Property:	Object_Identifier	= "Test_Loop"
Property:	Object_Type	= LOOP
Default Property:	Present_Value	= 8.3
Property:	Description	= "A simple test loop"
Property:	Status_Flags	= {FALSE,TRUE,FALSE,FALSE}
Property:	State	= NORMAL
Property:	Reliability	= RELIABLE
Property:	Out_Of_Service	= FALSE
Property:	Update_Interval	= 1
Property:	Output_Units	= "PSI"
Property:	Controlled_Variable_Reference	= ("1AH1DMPR", Present_Value)
Property:	Process_Variable_Reference	= ("1AH1MAT", Present_Value)
Property:	Process_Variable_Value	= 58.1
Property:	Process_Units	= "DEGF"
Property:	Setpoint_Reference	= ("MA_SetPoint", Present_Value)
Property:	Setpoint	= 58
Property:	Action	= DIRECT
Property:	Proportional_Constant	= 0.5
Property:	Proportional_Constant_Units	= "PSI/DEGF"
Property:	Integral_Constant	= 0.1
Property:	Integral_Constant_Units	= "1/MIN."
Property:	Derivative_Constant	= 0
Property:	Derivative_Constant_Units	= ""
Property:	Bias	= 9
Property:	Maximum_Output	= 15
Property:	Minimum_Output	= 3

Multi-State Input

Key Property:	Object_Identifier	= "Fan1_Input"
Property:	Object_Type	= MULTISTATE_INPUT
Default Property:	Present_Value	= STATE_2
Property:	Description	= "2-speed Fan#1"
Property:	Status_Flags	= {FALSE, TRUE, FALSE, FALSE}
Property:	State	= NORMAL
Property:	Reliability	= RELIABLE
Property:	Out_of_Service	= FALSE
Property:	Number_of_States	= 3
Property:	State_Text	= ("Off", "On_Low", "On_High")

Multi-State Output

Key Property:	Object_Identifier	= "Fan1_Output"
Property:	Object_Type	= MULTISTATE_OUTPUT
Default Property:	Present_Value	= STATE_2
Property:	Description	= "2-speed Fan#1"
Property:	Status_Flags	= {FALSE, TRUE, FALSE, FALSE}
Property:	State	= NORMAL
Property:	Reliability	= RELIABLE
Property:	Out_of_Service	= FALSE
Property:	Number_of_States	= 3
Property:	State_Text	= ("Off", "On_Low", "On_High")

Program

There is no standard configuration for this object because the underlying program and many of its properties are vendor-specific. The initial configuration must be supplied by the vendor.

Schedule

Key Property:	Object_Identifier	= "Rm208Sched"
Property:	Object_Type	= SCHEDULE
Default Property:	Present_Value	= ACTIVE
Property:	Effective_Period	= ((90/9/5),(91/6/10))
Property:	Monday_Schedule	= ((8:00,ACTIVE),(17:00,INACTIVE))
Property:	Tuesday_Schedule	= ((8:00,ACTIVE))
Property:	Wednesday_Schedule	= ((8:00,ACTIVE),(17:00,INACTIVE)),
Property:	Thursday_Schedule	= ((8:00,ACTIVE),(17:00,INACTIVE), (19:00,ACTIVE),(23:30,INACTIVE))
Property:	Friday_Schedule	= ((8:00,ACTIVE),(17:00,INACTIVE))
Property:	Saturday_Schedule	= ((10:00,ACTIVE),(17:00,INACTIVE))
Property:	Sunday_Schedule	= ((0:00,INACTIVE))
Property:	Exception_Schedule	= (((90/11/23),(0:00,INACTIVE),10), (HOLIDAYS,(0:00,INACTIVE),11), (((91/3/5),(91/3/7)), ((9:00,ACTIVE), (14:00,INACTIVE)),6))
Property:	Property_Reference	= (Rm208RTP,Present_Value)
Property:	Priority_For_Writing	= 15

5. TEST CASE NAMING CONVENTIONS

The abstract test cases are arranged in hierarchical structure of nested groups. Each test case is identified by a sequence of alpha-numeric fields separated by periods (.). Each field corresponds to a different level in the hierarchy. The last field is a sequential number which identifies this test case from all of the others in the same group. The result is an outline form which is illustrated in Figure 5.1.

A. Capability Test
   - A.1 Mandatory Object-Types[1]
   - A.2 Optional Object-Types
   - A.3 Property Functional Ranges

B. Behavior Tests: responses to valid behavior by peer implementation
   - B.1 Application Services Initiated by Tester
     - B.1.1 Alarm and Event Services
     - B.1.2 File Access Services
     - B.1.3 Object Access Services
     - B.1.4 Remote Device Management Services
     - B.1.5 Virtual Terminal Services
   - B.2 Application Services Initiated by Implementation Under Test
     - B.2.1 Alarm and Event Services
     - B.2.2 File Access Services
     - B.2.3 Object Access Services
     - B.2.4 Remote Device Management Services
     - B.2.5 Virtual Terminal Services

C. Behavior Tests: response to syntactically invalid behavior by peer implementation

D. Behavior Tests: response to syntactically correct but inopportune events by peer implementation.

Figure 5.1 Outline for Abstract Test Suite

Each test case begins with an explanation of the test purpose. This is followed by a brief description of the test and the expected results. The actions of the lower tester, upper tester and the IUT are then explained in detail. Finally the conditions for a pass result, a fail result, or an inconclusive result are described.

Test groups C and D are not included in this version of the abstract test suite. The current draft of BACnet defines a reject_PDU type which will be used in group C tests but the details of how a device should respond to invalid service requests (BACnet clause 6.2.1) have not been specified. There is a similar problem for the group D tests. Error-classes and error-codes have been defined but details about how they should be used are lacking.

---

[1] The current draft standard does not specify which object-types must be supported.

6. CAPABILITY TESTS

This group of abstract test cases are used to determine if BACnet standard object types are correctly supported. They are broken into two categories, tests to verify the existence of each object type and all of its properties, and tests for the range of values supported by the individual properties of the objects (functional range). To avoid unnecessary redundancy the functional range of a property is tested only once and it is assumed that all object types which have a property of that type correctly implement the functional range if the test is passed for one object type. These abstract tests make use of the ReadProperty, WriteProperty, and optionally the ReadMultipleProperty application services. It is assumed that the implementation of these services has been established prior to execution of these tests.

6.1 Object-type Support Tests

The SPC has not yet agreed on a policy regarding which standard object-types will be required in order to claim conformance to BACnet or if they will be grouped by conformance class. This test suite assumes that all object-types are optional. The test cases will be renumbered in a future version when the outstanding conformance issues are resolved by the committee (see Figure 5.1).

Test A.2.1 Support for the Analog Input Object-Type

Test Purpose:

Verify that the implementation supports the Analog Input object-type.

BACnet Clause Reference: 5.1

Test Description:

The lower tester issues several ReadProperty requests or a single ReadMultipleProperty request in an attempt to read all of the properties of the Analog Input object "1AH1MAT". The single ReadMultipleProperty request option is preferred if the ReadMultipleProperty service is supported by the IUT.

Expected Result:

A successful attempt to read all of the properties of "1AH1MAT". The values returned will be the values defined in the standard object configuration.

Lower Tester:

Option 1 - Issue a ReadMultipleProperty service request with the following argument:

'List of Read Access Specifications' = ("1AH1MAT", ALL)

Option 2 - Issue a sequence of ReadProperty requests, one for each of the sixteen properties of the object. The sequence of arguments is shown below.

'Read Access Specification' = ("1AH1MAT", Object_Identifier)
'Read Access Specification' = ("1AH1MAT", Object_Type)
'Read Access Specification' = ("1AH1MAT", Present_Value)
'Read Access Specification' = ("1AH1MAT", Description)
'Read Access Specification' = ("1AH1MAT", Device_Type)
'Read Access Specification' = ("1AH1MAT", Status_Flags)
'Read Access Specification' = ("1AH1MAT", State)
'Read Access Specification' = ("1AH1MAT", Reliability)

'Read Access Specification' = ("1AH1MAT", Out_of_Service)
'Read Access Specification' = ("1AH1MAT", Update_Interval)
'Read Access Specification' = ("1AH1MAT", Units)
'Read Access Specification' = ("1AH1MAT", Min_Pres)
'Read Access Specification' = ("1AH1MAT", Max_Pres)
'Read Access Specification' = ("1AH1MAT", Min_Raw)
'Read Access Specification' = ("1AH1MAT", Max_Raw)
'Read Access Specification' = ("1AH1MAT", Resolution)

IUT:

Process the ReadMultipleProperty request or the sequence of ReadProperty requests and issue a response containing the requested information.

Upper Tester:

No action is required for this test case.

Pass Results:

Option 1 - A Result(+) confirm primitive is received by the lower tester containing the following argument:

'List of Read Access Results' = ("1AH1MAT", ANALOG_INPUT, 58.1, "Mixed Air Temperature", "100 OHM RTD", {FALSE, TRUE, FALSE, FALSE}, NORMAL, RELIABLE, FALSE, 10, "DEGF", -50, 250, 0, 4095, 0.1)

Option 2 - A sequence of Result(+) confirm primitives are received by the lower tester, one for each request issued. Each Result(+) primitive will contain one argument. The sequence will be:

'Read Access Result' = "1AHU1MAT"
'Read Access Result' = ANALOG_INPUT
'Read Access Result' = 58.1
'Read Access Result' = "Mixed Air Temperature"
'Read Access Result' = "100 OHM RTD"
'Read Access Result' = {FALSE, TRUE, FALSE, FALSE}
'Read Access Result' = NORMAL
'Read Access Result' = RELIABLE
'Read Access Result' = FALSE
'Read Access Result' = 10
'Read Access Result' = "DEGF"
'Read Access Result' = -50
'Read Access Result' = 250
'Read Access Result' = 0
'Read Access Result' = 4095
'Read Access Result' = 0.1

Fail Results:

This test fails if a Result(-) confirm primitive is received by the lower tester with an error class of OBJECT or an error class of PROPERTY. The test also fails if one or more of the property values returned in the Result(+) primitives is not correct. In this case a configuration error may be the cause and this fact should be noted in the test report.

Inconclusive Results:

The test results are inconclusive if a Result(-) confirm primitive is received by the lower tester with any error class other than OBJECT or PROPERTY. The results are also inconclusive if a reply is not received to one or more of the read requests.

Test A.2.2 Support for the Analog Output Object-Type

Test Purpose:

Verify that the implementation supports the Analog Output object-type.

BACnet Clause Reference: 5.2

Test Description:

The lower tester issues several ReadProperty requests or a single ReadMultipleProperty request in an attempt to read all of the properties of the Analog Output object "1AH1DMPR". The single ReadMultipleProperty request option is preferred if the ReadMultipleProperty service is supported by the IUT.

Expected Result:

A successful attempt to read all of the properties of "1AH1DMPR". The values returned will be the values defined in the standard object configuration.

Lower Tester:

Option 1 - Issue a ReadMultipleProperty service request with the following argument:

'List of Read Access Specifications' = ("1AH1DMPR", ALL)

Option 2 - Issue a sequence of ReadProperty requests, one for each of the sixteen properties of the object. The sequence of arguments is shown below.

'Read Access Specification' = ("1AH1DMPR", Object_Identifier)
'Read Access Specification' = ("1AH1DMPR", Object_Type)
'Read Access Specification' = ("1AH1DMPR", Present_Value)
'Read Access Specification' = ("1AH1DMPR", Description)
'Read Access Specification' = ("1AH1DMPR", Device_Type)
'Read Access Specification' = ("1AH1DMPR", Status_Flags)
'Read Access Specification' = ("1AH1DMPR", State)
'Read Access Specification' = ("1AH1DMPR", Reliability)
'Read Access Specification' = ("1AH1DMPR", Out_of_Service)
'Read Access Specification' = ("1AH1DMPR", Update_Interval)
'Read Access Specification' = ("1AH1DMPR", Units)
'Read Access Specification' = ("1AH1DMPR", Min_Pres)
'Read Access Specification' = ("1AH1DMPR", Max_Pres)
'Read Access Specification' = ("1AH1DMPR", Min_Raw)
'Read Access Specification' = ("1AH1DMPR", Max_Raw)
'Read Access Specification' = ("1AH1DMPR", Resolution)

IUT:

Process the ReadMultipleProperty request or the sequence of ReadProperty requests and issue a response containing the requested information.

Upper Tester:

No action is required for this test case.

Pass Results:

Option 1 - A Result(+) confirm primitive is received by the lower tester containing the following argument:

'List of Read Access Results' = ("1AH1DMPR", ANALOG_OUTPUT, 75.0, "Damper Actuator", "3-8 PSI Actuator", {FALSE, TRUE, FALSE, FALSE}, NORMAL, RELIABLE, FALSE, 10, "%", 0, 100, 0, 4095, 0.1)

Option 2 - A sequence of Result(+) confirm primitives are received by the lower tester, one for each request issued. Each Result(+) primitive will contain one argument. The sequence will be:

'Read Access Result' = "1AH1DMPR"
'Read Access Result' = ANALOG_OUTPUT
'Read Access Result' = 75.0
'Read Access Result' = "Damper Actuator"
'Read Access Result' = "3-8 PSI Actuator"
'Read Access Result' = {FALSE, TRUE, FALSE, FALSE}
'Read Access Result' = NORMAL
'Read Access Result' = RELIABLE
'Read Access Result' = FALSE
'Read Access Result' = 10
'Read Access Result' = "%"
'Read Access Result' = 0
'Read Access Result' = 100
'Read Access Result' = 0
'Read Access Result' = 4095
'Read Access Result' = 0.1

Fail Results:

This test fails if a Result(-) confirm primitive is received by the lower tester with an error class of OBJECT or an error class of PROPERTY. The test also fails if one or more of the property values returned in the Result(+) primitives is not correct. In this case a configuration error may be the cause and this fact should be noted in the test report.

Inconclusive Results:

The test results are inconclusive if a Result(-) confirm primitive is received by the lower tester with any error class other than OBJECT or PROPERTY. The results are also inconclusive if a reply is not received to one or more of the read requests.

Test A.2.3 Support for the Analog Value Object-Type

Test Purpose:

Verify that the implementation supports the Analog Value object-type.

BACnet Clause Reference: 5.3

Test Description:

The lower tester issues several ReadProperty requests or a single ReadMultipleProperty request in an

attempt to read all of the properties of the Analog Value object "MA_SetPoint". The single ReadMultipleProperty request option is preferred if the ReadMultipleProperty service is supported by the IUT.

Expected Result:

A successful attempt to read all of the properties of "MA_SetPoint". The values returned will be the values defined in the standard object configuration.

Lower Tester:

Option 1 - Issue a ReadMultipleProperty service request with the following argument:

'List of Read Access Specifications' = ("MA_SetPoint", ALL)

Option 2 - Issue a sequence of ReadProperty requests, one for each of the nine properties of the object. The sequence of arguments is shown below.

'Read Access Specification' = ("MA_SetPoint", Object_Identifier)
'Read Access Specification' = ("MA_SetPoint", Object_Type)
'Read Access Specification' = ("MA_SetPoint", Present_Value)
'Read Access Specification' = ("MA_SetPoint", Description)
'Read Access Specification' = ("MA_SetPoint", Status_Flags)
'Read Access Specification' = ("MA_SetPoint", State)
'Read Access Specification' = ("MA_SetPoint", Reliability)
'Read Access Specification' = ("MA_SetPoint", Out_of_Service)
'Read Access Specification' = ("MA_SetPoint", Units)

IUT:

Process the ReadMultipleProperty request or the sequence of ReadProperty requests and issue a response containing the requested information.

Upper Tester:

No action is required for this test case.

Pass Results:

Option 1 - A Result(+) confirm primitive is received by the lower tester containing the following argument:

'List of Read Access Results' = ("MA_SetPoint", ANALOG_VALUE, 58.0, "Loop Set Point", {FALSE, TRUE, FALSE, FALSE}, NORMAL, RELIABLE, FALSE, "DEGF")

Option 2 - A sequence of Result(+) confirm primitives are received by the lower tester, one for each request issued. Each Result(+) primitive will contain one argument. The sequence will be:

'Read Access Result' = "MA_SetPoint"
'Read Access Result' = ANALOG_VALUE
'Read Access Result' = 58.0
'Read Access Result' = "Loop Set Point"
'Read Access Result' = {FALSE, TRUE, FALSE, FALSE}
'Read Access Result' = NORMAL
'Read Access Result' = RELIABLE
'Read Access Result' = FALSE
'Read Access Result' = "DEGF"

Fail Results:

This test fails if a Result(-) confirm primitive is received by the lower tester with an error class of OBJECT or an error class of PROPERTY. The test also fails if one or more of the property values returned in the Result(+) primitives is not correct. In this case a configuration error may be the cause and this fact should be noted in the test report.

Inconclusive Results:

The test results are inconclusive if a Result(-) confirm primitive is received by the lower tester with any error class other than OBJECT or PROPERTY. The results are also inconclusive if a reply is not received to one or more of the read requests.

Test A.2.4 Support for the Binary Input Object-Type

Test Purpose:

Verify that the implementation supports the Binary Input object-type.

BACnet Clause Reference: 5.4

Test Description:

The lower tester issues several ReadProperty requests or a single ReadMultipleProperty request in an attempt to read all of the properties of the Binary Input object "HighPressSwitch". The single ReadMultipleProperty request option is preferred if the ReadMultipleProperty service is supported by the IUT.

Expected Result:

A successful attempt to read all of the properties of "HighPressSwitch". The values returned will be the values defined in the standard object configuration.

Lower Tester:

Option 1 - Issue a ReadMultipleProperty service request with the following argument:

'List of Read Access Specifications' = ("HighPressSwitch", ALL)

Option 2 - Issue a sequence of ReadProperty requests, one for each of the fifteen properties of the object. The sequence of arguments is shown below.

'Read Access Specification' = ("HighPressSwitch",Object_Identifier)
'Read Access Specification' = ("HighPressSwitch", Object_Type)
'Read Access Specification' = ("HighPressSwitch", Description)
'Read Access Specification' = ("HighPressSwitch", Present_Value)
'Read Access Specification' = ("HighPressSwitch", Polarity)
'Read Access Specification' = ("HighPressSwitch", Inactive_Text)
'Read Access Specification' = ("HighPressSwitch", Active_Text)
'Read Access Specification' = ("HighPressSwitch", Status_Flags)
'Read Access Specification' = ("HighPressSwitch", State)
'Read Access Specification' = ("HighPressSwitch", Reliability)
'Read Access Specification' = ("HighPressSwitch", Out_of_Service)
'Read Access Specification' = ("HighPressSwitch", Change_of_State_Time)
'Read Access Specification' = ("HighPressSwitch", Elapsed_Active_Time)
'Read Access Specification' = ("HighPressSwitch", Change_of_State_Count)
'Read Access Specification' = ("HighPressSwitch", Time_of_Reset)

IUT:

Process the ReadMultipleProperty request or the sequence of ReadProperty requests and issue a response containing the requested information.

Upper Tester:

No action is required for this test case.

Pass Results:

Option 1 - A Result(+) confirm primitive is received by the lower tester containing the following argument:

'List of Read Access Results' = ("HighPressSwitch", BINARY_INPUT, "Penthouse Supply High Static", ACTIVE, NORMAL, "Static Pressure OK", "High Pressure Alarm", {FALSE, TRUE, FALSE, FALSE}, NORMAL, RELIABLE, FALSE, (90/03/23, 6, 19:01:34), 401, 134, (90/01/01, 2, 00:00:00))

Option 2 - A sequence of Result(+) confirm primitives are received by the lower tester, one for each request issued. Each Result(+) primitive will contain one argument. The sequence will be:

'Read Access Result' = "HighPressSwitch"
'Read Access Result' = BINARY_INPUT
'Read Access Result' = "Penthouse Supply High Static"
'Read Access Result' = ACTIVE
'Read Access Result' = NORMAL;
'Read Access Result' = "Static Pressure OK"
'Read Access Result' = "High Pressure Alarm"
'Read Access Result' = {FALSE, TRUE, FALSE, FALSE}
'Read Access Result' = NORMAL
'Read Access Result' = RELIABLE
'Read Access Result' = FALSE
'Read Access Result' = (90/03/23, 6, 19:01:34)
'Read Access Result' = 401
'Read Access Result' = 134
'Read Access Result' = (90/01/01, 2, 00:00:00)

Fail Results:

This test fails if a Result(-) confirm primitive is received by the lower tester with an error class of OBJECT or an error class of PROPERTY. The test also fails if one or more of the property values returned in the Result(+) primitives is not correct. In this case a configuration error may be the cause and this fact should be noted in the test report.

Inconclusive Results:

The test results are inconclusive if a Result(-) confirm primitive is received by the lower tester with any error class other than OBJECT or PROPERTY. The results are also inconclusive if a reply is not received to one or more of the read requests.

Test A.2.5 Support for the Binary Output Object-Type

Test Purpose:

Verify that the implementation supports the Binary Output object-type.

BACnet Clause Reference 5.5

Test Description:

The lower tester issues several ReadProperty requests or a single ReadMultipleProperty request in an attempt to read all of the properties of the Binary Output object "Floor3ExhaustFan". The single ReadMultipleProperty request option is preferred if the ReadMultipleProperty service is supported by the IUT.

Expected Result:

A successful attempt to read all of the properties of "Floor3ExhaustFan". The values returned will be the values defined in the standard object configuration.

Lower Tester:

Option 1 - Issue a ReadMultipleProperty service request with the following argument:

'List of Read Access Specifications' = ("Floor3ExhaustFan", ALL)

Option 2 - Issue a sequence of ReadProperty requests, one for each of the seventeen properties of the object. The sequence of arguments is shown below.

'Read Access Specification' = ("Floor3ExhaustFan",Object_Identifier)
'Read Access Specification' = ("Floor3ExhaustFan", Object_Type)
'Read Access Specification' = ("Floor3ExhaustFan", Description)
'Read Access Specification' = ("Floor3ExhaustFan", Present_Value)
'Read Access Specification' = ("Floor3ExhaustFan", Polarity)
'Read Access Specification' = ("Floor3ExhaustFan", Inactive_Text)
'Read Access Specification' = ("Floor3ExhaustFan", Active_Text)
'Read Access Specification' = ("Floor3ExhaustFan", Status_Flags)
'Read Access Specification' = ("Floor3ExhaustFan", State)
'Read Access Specification' = ("Floor3ExhaustFan", Reliability)
'Read Access Specification' = ("Floor3ExhaustFan", Out_of_Service)
'Read Access Specification' = ("Floor3ExhaustFan", Change_of_State_Time)
'Read Access Specification' = ("Floor3ExhaustFan", Run_Time)
'Read Access Specification' = ("Floor3ExhaustFan", Change_of_State_Count)
'Read Access Specification' = ("Floor3ExhaustFan", Time_of_Reset)
'Read Access Specification' = ("Floor3ExhaustFan", Minimum_Off_Time)
'Read Access Specification' = ("Floor3ExhaustFan", Minimum_On_Time)

IUT:

Process the ReadMultipleProperty request or the sequence of ReadProperty requests and issue a response containing the requested information.

Upper Tester:

No action is required for this test case.

Pass Results:

Option 1 - A Result(+) confirm primitive is received by the lower tester containing the following argument:

'List of Read Access Results' = ("Floor3ExhaustFan", BINARY_OUTPUT, "Third floor bathroom exhaust fan", INACTIVE, REVERSE, "Fan is turned off", "Fan is running", {FALSE, TRUE, FALSE, FALSE}, NORMAL, RELIABLE, FALSE, (90/03/23, 6, 19:01:34), 401, 134, (90/01/01, 2, 00:00:00),

100, 10)

Option 2 - A sequence of Result(+) confirm primitives are received by the lower tester, one for each request issued. Each Result(+) primitive will contain one argument. The sequence will be:

'Read Access Result' = "Floor3ExhaustFan"
'Read Access Result' = BINARY_OUTPUT
'Read Access Result' = "Third Floor bathroom exhaust fan"
'Read Access Result' = INACTIVE
'Read Access Result' = REVERSE;
'Read Access Result' = "Fan is turned off"
'Read Access Result' = "Fan is running"
'Read Access Result' = {FALSE, TRUE, FALSE, FALSE}
'Read Access Result' = NORMAL
'Read Access Result' = RELIABLE
'Read Access Result' = FALSE
'Read Access Result' = (90/03/23, 6, 19:01:34)
'Read Access Result' = 401
'Read Access Result' = 134
'Read Access Result' = (90/01/01, 2, 00:00:00)
'Read Access Result' = 100
'Read Access Result' = 10

Fail Results:

This test fails if a Result(-) confirm primitive is received by the lower tester with an error class of OBJECT or an error class of PROPERTY. The test also fails if one or more of the property values returned in the Result(+) primitives is not correct. In this case a configuration error may be the cause and this fact should be noted in the test report.

Inconclusive Results:

The test results are inconclusive if a Result(-) confirm primitive is received by the lower tester with any error class other than OBJECT or PROPERTY. The results are also inconclusive if a reply is not received to one or more of the read requests.

Test A.2.6 Support for the Binary Value Object-Type

Test Purpose:

Verify that the implementation supports the Binary Value object-type.

BACnet Clause Reference: 5.6

Test Description:

The lower tester issues several ReadProperty requests or a single ReadMultipleProperty request in an attempt to read all of the properties of the Binary Value object "ExhaustFanEnable". The single ReadMultipleProperty request option is preferred if the ReadMultipleProperty service is supported by the IUT.

Expected Result:

A successful attempt to read all of the properties of "ExhaustFanEnable". The values returned will be the values defined in the standard object configuration.

Lower Tester:

Option 1 - Issue a ReadMultipleProperty service request with the following argument:

'List of Read Access Specifications' = ("ExhaustFanEnable", ALL)

Option 2 - Issue a sequence of ReadProperty requests, one for each of the sixteen properties of the object. The sequence of arguments is shown below.

'Read Access Specification' = ("ExhaustFanEnable",Object_Identifier)
'Read Access Specification' = ("ExhaustFanEnable", Object_Type)
'Read Access Specification' = ("ExhaustFanEnable", Description)
'Read Access Specification' = ("ExhaustFanEnable", Present_Value)
'Read Access Specification' = ("ExhaustFanEnable", Inactive_Text)
'Read Access Specification' = ("ExhaustFanEnable", Active_Text)
'Read Access Specification' = ("ExhaustFanEnable", Status_Flags)
'Read Access Specification' = ("ExhaustFanEnable", State)
'Read Access Specification' = ("ExhaustFanEnable", Reliability)
'Read Access Specification' = ("ExhaustFanEnable", Out_of_Service)
'Read Access Specification' = ("ExhaustFanEnable", Change_of_State_Time)
'Read Access Specification' = ("ExhaustFanEnable", Elapsed_Active_Time)
'Read Access Specification' = ("ExhaustFanEnable", Change_of_State_Count)
'Read Access Specification' = ("ExhaustFanEnable", Time_of_Reset)
'Read Access Specification' = ("ExhaustFanEnable", Minimum_Off_Time)
'Read Access Specification' = ("ExhaustFanEnable", Minimum_On_Time)

IUT:

Process the ReadMultipleProperty request or the sequence of ReadProperty requests and issue a response containing the requested information.

Upper Tester:

No action is required for this test case.

Pass Results:

Option 1 - A Result(+) confirm primitive is received by the lower tester containing the following argument:

'List of Read Access Results' = ("ExhaustFanEnable", BINARY_VALUE, "Exhaust Fan Operator Enable", ACTIVE, "Enabled by Operator", "Fan Not Enabled by Operator", {FALSE, TRUE, FALSE, FALSE}, NORMAL, RELIABLE, FALSE, (90/03/23, 6, 19:01:34), 401, 134, (90/01/01, 2, 00:00:00), 0, 0)

Option 2 - A sequence of Result(+) confirm primitives are received by the lower tester, one for each request issued. Each Result(+) primitive will contain one argument. The sequence will be:

'Read Access Result' = "ExhaustFanEnable"
'Read Access Result' = BINARY_VALUE
'Read Access Result' = "Exhaust Fan Operator Enable"
'Read Access Result' = ACTIVE
'Read Access Result' = "Enabled by Operator"
'Read Access Result' = "Fan Not Enabled by Operator"
'Read Access Result' = {FALSE, TRUE, FALSE, FALSE}
'Read Access Result' = NORMAL

'Read Access Result' = RELIABLE
'Read Access Result' = FALSE
'Read Access Result' = (90/03/23, 6, 19:01:34)
'Read Access Result' = 401
'Read Access Result' = 134
'Read Access Result' = (90/01/01, 2, 00:00:00)
'Read Access Result' = 0
'Read Access Result' = 0

Fail Results:

This test fails if a Result(-) confirm primitive is received by the lower tester with an error class of OBJECT or an error class of PROPERTY. The test also fails if one or more of the property values returned in the Result(+) primitives is not correct. In this case a configuration error may be the cause and this fact should be noted in the test report.

Inconclusive Results:

The test results are inconclusive if a Result(-) confirm primitive is received by the lower tester with any error class other than OBJECT or PROPERTY. The results are also inconclusive if a reply is not received for one or more of the read requests.

Test A.2.7 Support for the Calendar Object-Type

Test Purpose:

Verify that the implementation supports the Calendar object-type.

BACnet Clause Reference: 5.7

Test Description:

The lower tester issues several ReadProperty requests or a single ReadMultipleProperty request in an attempt to read all of the properties of the Calendar object "HOLIDAYS". The single ReadMultipleProperty request option is preferred if the ReadMultipleProperty service is supported by the IUT.

Expected Result:

A successful attempt to read all of the properties of "HOLIDAYS". The values returned will be the values defined in the standard object configuration.

Lower Tester:

Option 1 - Issue a ReadMultipleProperty service request with the following argument:

'List of Read Access Specifications' = ("HOLIDAYS", ALL)

Option 2 - Issue a sequence of ReadProperty requests, one for each of the four properties of the object. The sequence of arguments is shown below.

'Read Access Specification' = ("HOLIDAYS",Object_Identifier)
'Read Access Specification' = ("HOLIDAYS", Object_Type)
'Read Access Specification' = ("HOLIDAYS", Present_Value)
'Read Access Specification' = ("HOLIDAYS", DateList)

IUT:

Process the ReadMultipleProperty request or the sequence of ReadProperty requests and issue a response containing the requested information.

Upper Tester:

No action is required for this test case.

Pass Results:

Option 1 - A Result(+) confirm primitive is received by the lower tester containing the following argument:

'List of Read Access Results' = ("HOLIDAYS", CALENDAR, ACTIVE, ((90/2/19), (90/5/28), ((90/12/24), (91/1/4)))

Option 2 - A sequence of Result(+) confirm primitives are received by the lower tester, one for each request issued. Each Result(+) primitive will contain one argument. The sequence will be:

'Read Access Result' = "HOLIDAYS"
'Read Access Result' = CALENDAR
'Read Access Result' = ACTIVE
'Read Access Result' = ((90/2/19), (90/5/28), ((90/12/24), (91/1/4)))

Fail Results:

This test fails if a Result(-) confirm primitive is received by the lower tester with an error class of OBJECT or an error class of PROPERTY. The test also fails if one or more of the property values returned in the Result(+) primitives is not correct. In this case a configuration error may be the cause and this fact should be noted in the test report.

Inconclusive Results:

The test results are inconclusive if a Result(-) confirm primitive is received by the lower tester with any error class other than OBJECT or PROPERTY. The results are also inconclusive if a reply is not received for one or more of the read requests.

Test A.2.8 Support for the Command Object-Type

Test Purpose:

Verify that the implementation supports the Command object-type.

BACnet Clause Reference: 5.8

Test Description:

The lower tester issues several ReadProperty requests or a single ReadMultipleProperty request in an attempt to read all of the properties of the Command object "CommandTest". The single ReadMultipleProperty request option is preferred if the ReadMultipleProperty service is supported by the IUT.

If the first part is successful the lower tester issues several WriteProperty requests, each one writing a new value to the Present_Value property of "ConformTest". A series of ReadProperty requests is issued after each WriteProperty request is confirmed to determine if the "command" functions of the object have been implemented. For the special cases where the StateN property indicates that the properties to be "commanded" reside in the lower tester subsequent ReadProperty requests are not needed.

Expected Result:

A successful attempt to read all of the properties of "CommandTest". The values returned will be the values determined in the standard configuration. This is followed by successfully "commanding" the object into each of its possible states.

Lower Tester:

Step 1, Option 1 - Issue a ReadMultipleProperty service request with the following argument:

'List of Read Access Results' = ("CommandTest", ALL)

Step 1, Option 2 - Issue a sequence of ReadProperty service requests, one for each of the eleven properties of the object. the sequence of arguments is shown below.

'Read Access Specification' = ("CommandTest", Object_Identifier)
'Read Access Specification' = ("CommandTest", Object_Type)
'Read Access Specification' = ("CommandTest", Present_Value)
'Read Access Specification' = ("CommandTest", Description)
'Read Access Specification' = ("CommandTest", State0)
'Read Access Specification' = ("CommandTest", State1)
'Read Access Specification' = ("CommandTest", State2)
'Read Access Specification' = ("CommandTest", State3)
'Read Access Specification' = ("CommandTest", State4)
'Read Access Specification' = ("CommandTest", State5)
'Read Access Specification' = ("CommandTest", State6)
'Read Access Specification' = ("CommandTest", State7)

Step 2 - Issue a WriteProperty service request with the following argument:

'Write Access Specification' = ("CommandTest", Present_Value, 0)

When a confirmation is received (possibly after retries) issue ReadProperty service requests with the following arguments.

'Read Access Specification' = ("CommandTest", Present_Value)
'Read Access Specification' = ("1AH1MAT", Min_Raw)
'Read Access Specification' = ("1AH1DMPR", Units)
'Read Access Specification' = ("1AH1DMPR", Present_Value)

Step 3 - Issue a WriteProperty service request with the following argument:

'Write Access Specification' = ("CommandTest", Present_Value, 1)
When a confirmation is received (possibly after retries) issue ReadProperty service requests with the following arguments.

'Read Access Specification' = ("CommandTest", Present_Value)
'Read Access Specification' = ("1AH1MAT", Min_Raw)
'Read Access Specification' = ("1AH1DMPR", Units)
'Read Access Specification' = ("1AH1DMPR", Present_Value)

Step 4 - Issue a WriteProperty service request with the following argument:

Write Access Specification = ("CommandTest", Present_Value, 2)

When a confirmation is received (possibly after retries) issue ReadProperty service requests with the following arguments.

'Read Access Specification' = ("CommandTest", Present_Value)
'Read Access Specification' = ("Floor3ExhaustFan", Present_Value)

Step 5 - Issue a WriteProperty service request with the following argument:

'Write Access Specification' = ("CommandTest", Present_Value, 3)

When a confirmation is received (possibly after retries) issue ReadProperty service requests with the following arguments.

'Read Access Specification' = ("CommandTest", Present_Value)
'Read Access Specification' = ("Floor3ExhaustFan", Present_Value)

Step 6 - Issue a WriteProperty service request with the following argument:

Write Access Specification = ("CommandTest", Present_Value, 4)

When a confirmation is received (possibly after retries) issue ReadProperty service requests with the following arguments.

'Read Access Specification' = ("CommandTest", Present_Value)
'Read Access Specification' = ("1AH1MA_SetPoint", Present,Value)
'Read Access Specification' = ("Floor3ExhaustFan", Present_Value)
'Read Access Specification' = ("MA_SetPoint", Present_Value)

Step 7 - Issue a WriteProperty service request with the following argument:

'Write Access Specification' = ("CommandTest", Present_Value, 5)

When a confirmation is received (possibly after retries) issue ReadProperty service requests with the following arguments.

'Read Access Specification' = ("CommandTest", Present_Value)
'Read Access Specification' = ("1AH1MA_SetPoint", Present,Value)
'Read Access Specification' = ("Floor3ExhaustFan", Present_Value)
'Read Access Specification' = ("MA_SetPoint", Present_Value)

Step 8 - Issue a WriteProperty service request with the following argument.

'Write Access Specification' = ("CommandTest", Present_Value, 6)

Step 9 - Issue a WriteProperty service request with the following argument.

'Write Access Specification' = ("CommandTest", Present_Value, 7)

IUT:

Step 1 - Process the ReadMultipleProperty request or the sequence of ReadProperty requests and issue a response containing the requested information.

Steps 2 - 9 When the WriteProperty service indication is received, change the value of the Present_Value property of the "CommandTest" object, carry out the implied command, and issue a Result(+) response primitive. When the subsequent ReadProperty service indications are received, issue a Result(+) response primitive conveying the requested values.

Upper Tester:

No action is required for this test.

Pass Result:

Step 1, Option 1 - A Result(+) confirm primitive is received by the lower tester containing the following argument:

'List of Read Access Results' = ("CommandTest", COMMAND, "Command Object used for Test Purposes", 0, ((NULL,"1AH1MAT",Min_Raw,100), (NULL,"1AH1DMPR",Units,"Percent"), (NULL, "1AH1DMPR", Present_Value, 61)), ((NULL,"1AH1MAT",Min_Raw,0), (NULL,"1AH1DMPR",Units,"%"), (NULL, "1AH1DMPR", Present_Value, 75)), (NULL, "Floor3ExhaustFan", Present_Value, ACTIVE), (NULL, "Floor3ExhaustFan", Present_Value, INACTIVE), ((NULL,"1AH1DMPR",NULL,20), (NULL,"Floor3ExhaustFan",NULL, ACTIVE), (NULL, "MA_SetPoint", NULL, 76)), ((NULL,"1AH1DMPR",NULL,75), (NULL,"Floor3ExhaustFan",NULL,INACTIVE),(NULL, "MA_SetPoint",NULL, 58)),((lower tester, "TestLoop", Process_Variable_Reference, ("MA_SetPoint", Present_Value)), (lower tester, "Rm208Sched", Sunday_Schedule, (0:00, INACTIVE), (8:00, ACTIVE), (2:00, INACTIVE)), (lower tester, "Fan1_Output", NULL, State3)), ((lower tester, "TestLoop", Process_Variable_Reference, ("1AH1MAT", Present_Value)), (lower tester, "Rm208Sched", Sunday_Schedule, (0:00, INACTIVE)), (lower tester, "Fan1_Output", NULL, State2))

Step 1, Option 2 - A sequence of Result(+) confirm primitives are received by the lower tester, one for each request issued. Each Result(+) primitive will contain one argument. The sequence will be.

'Read Access Result' = "CommandTest"
'Read Access Result' = COMMAND
'Read Access Result' = "Command Object used for Test Purposes"
'Read Access Result' = 0
'Read Access Result' = ((NULL,"1AH1MAT",Min_Raw,100), (NULL,"1AH1DMPR",Units,"Percent"), (NULL, "1AH1DMPR", Present_Value, 61))
'Read Access Result' = ((NULL,"1AH1MAT",Min_Raw,0), (NULL,"1AH1DMPR",Units,"%"), (NULL, "1AH1DMPR", Present_Value, 75))
'Read Access Result' = (NULL, "Floor3ExhaustFan", Present_Value, ACTIVE)
'Read Access Result' = (NULL, "Floor3ExhaustFan", Present_Value, INACTIVE)
'Read Access Result' = ((NULL,"1AH1DMPR",NULL,20), (NULL,"Floor3ExhaustFan",NULL, ACTIVE), (NULL, "MA_SetPoint", NULL, 76))
'Read Access Result' = ((NULL,"1AH1DMPR",NULL,75), (NULL,"Floor3ExhaustFan",NULL, INACTIVE), (NULL, "MA_SetPoint", NULL, 58))
'Read Access Result' = ((lower tester, "TestLoop", Process_Variable_Reference, ("MA_SetPoint", Present_Value)), (lower tester, "Rm208Sched", Sunday_Schedule, (0:00, INACTIVE), (8:00, ACTIVE), (2:00, INACTIVE)), (lower tester, "Fan1_Output", NULL, State3))
'Read Access Result' = ((lower tester, "TestLoop", Process_Variable_Reference, ("1AH1MAT",

Present_Value)), (lower tester, "Rm208Sched", Sunday_Schedule, (0:00, INACTIVE)), (lower tester, "Fan1_Output", NULL, State2))

Step 2 - A Result(+) confirm primitive is received by the lower tester in response to the WriteProperty service request. A sequence of Result(+) confirm primitives is received by the lower tester , one for each of the ReadProperty service requests. Each Result(+) primitive conveys one argument. The sequence will be:

'Read Access Result' = ("CommandTest", Present_Value, 0)
'Read Access Result' = ("1AH1MAT", Min_Raw, 100)
'Read Access Result' = ("1AH1DMPR", Units, "Percent")
'Read Access Result' = ("1AH1DMPR", Present_Value, 61)

Step 3 - A Result(+) confirm primitive is received by the lower tester in response to the WriteProperty service request. A sequence of Result(+) confirm primitives is received by the lower tester , one for each of the ReadProperty service requests. Each Result(+) primitive conveys one argument. The sequence will be:

'Read Access Result' = ("CommandTest", Present_Value, 1)
'Read Access Result' = ("1AH1MAT", Min_Raw, 100)
'Read Access Result' = ("1AH1DMPR", Units, "%")
'Read Access Result' = ("1AH1DMPR", Present_Value, 75)

Step 4 - A Result(+) confirm primitive is received by the lower tester in response to the WriteProperty service request. A sequence of Result(+) confirm primitives is received by the lower tester , one for each of the ReadProperty service requests. Each Result(+) primitive conveys one argument. The sequence will be:

'Read Access Result' = ("CommandTest", Present_Value, 2)
'Read Access Result' = ("Floor3ExhaustFan", Present_Value, ACTIVE)

Step 5 - A Result(+) confirm primitive is received by the lower tester in response to the WriteProperty service request. A sequence of Result(+) confirm primitives is received by the lower tester , one for each of the ReadProperty service requests. Each Result(+) primitive conveys one argument. The sequence will be:

'Read Access Result' = ("CommandTest", Present_Value, 3)
'Read Access Result' = ("Floor3ExhaustFan", Present_Value, INACTIVE)

Step 6 - A Result(+) confirm primitive is received by the lower tester in response to the WriteProperty service request. A sequence of Result(+) confirm primitives is received by the lower tester , one for each of the ReadProperty service requests. Each Result(+) primitive conveys one argument. The sequence will be:

'Read Access Result' = ("CommandTest", Present_Value, 4)
'Read Access Result' = ("1AH1DMPR", Present_Value, 20)
'Read Access Result' = ("Floor3ExhaustFan", Present_Value, ACTIVE)
'Read Access Result' = ("MA_SetPoint", Present_Value, 76)

Step 7 - A Result(+) confirm primitive is received by the lower tester in response to the WriteProperty service request. A sequence of Result(+) confirm primitives is received by the lower tester , one for each of the ReadProperty service requests. Each Result(+) primitive conveys one argument. The sequence will

be:

'Read Access Result' = ("CommandTest", Present_Value, 5)
'Read Access Result' = ("1AH1DMPR", Present_Value, 75)
'Read Access Result' = ("Floor3ExhaustFan", Present_Value, INACTIVE)
'Read Access Result' = ("MA_SetPoint", Present_Value, 58)

Step 8 - A Result(+) confirm primitive is received by the lower tester in response to the WriteProperty service request. A sequence of three WriteProperty service indication are received by the lower tester. The arguments conveyed by these service primitives are:

'Write Access Specification' = (NULL, "TestLoop", Process_Variable_Reference, ("1AH1MAT", Present_Value))
'Write Access Specification' = (NULL, "Rm208Sched", Sunday_Schedule, ((0:00, INACTIVE), (8:00, Active), (2:00, INACTIVE))
'Write Access Specification' = (NULL, "Fan1_Output", Present_Value, State3)

These WriteProperty service indications may appear in any order and the Result(+) confirm primitive may be interleaved with these WriteProperty service indications in any order.

Step 9 - A Result(+) confirm primitive is received by the lower tester in response to the WriteProperty service request. A sequence of three WriteProperty service indication are received by the lower tester. The arguments conveyed by these service primitives are:

'Write Access Specification' = (NULL, "TestLoop", Process_Variable_Reference, ("1AH1MAT", Present_Value))
'Write Access Specification' = (NULL, "Rm208Sched", Sunday_Schedule, (0:00, INACTIVE))
'Write Access Specification' = (NULL, "Fan1_Output", Present_Value, State3)

These WriteProperty service indications may appear in any order and the Result(+) confirm primitive may be interleaved with these WriteProperty service indications in any order.

Fail Result:

If the Result(+) confirm primitive received in response to the ReadProperty service requests in steps 2 - 7 do not return the specified value and there are no other algorithms running in the device which can change the values of these properties, then the test fails. In steps 8 and 9, if the WriteProperty service indications are not received or are incorrect then the test fails.

Inconclusive Results:

If the Result(+) confirm primitive received in response to the ReadProperty service requests in steps 2 - 7 do not return the specified value but there are other algorithms running in the device which can change the values of these properties, then the test is inconclusive. It is not clear if the Command object is not properly implemented or if the property values were changed by another algorithm before they could be read.

Test A.2.9 Support for the Device Object-Type

Test Purpose:

Verify that the implementation supports the Device object-type.

BACnet Clause Reference: 5.9

Test Description:

The lower tester issues several ReadProperty requests or a single ReadMultipleProperty request in an attempt to read all of the properties of the Device object specified in the PICS. The single ReadMultipleProperty request option is preferred if the ReadMultipleProperty service is supported by the IUT.

Expected Result:

A successful attempt to read all of the properties of the device object. Since the device object is device specific, the values returned will be the values defined in the PICS.

Lower Tester:

Option 1 - Issue a ReadMultipleProperty service request with the following argument:

'List of Read Access Specifications' = (device object ID, ALL)

Option 2 - Issue a sequence of ReadProperty requests, one for each of the nineteen properties of the object. The sequence of arguments is shown below.

'Read Access Specification' = (device object ID, Object_Identifier)
'Read Access Specification' = (device object ID, Object_Type)
'Read Access Specification' = (device object ID, System_Status)
'Read Access Specification' = (device object ID, Vendor_Name)
'Read Access Specification' = (device object ID, Model_Name)
'Read Access Specification' = (device object ID, Firmware_Revision)
'Read Access Specification' = (device object ID, Application_Software_Version)
'Read Access Specification' = (device object ID, Location)
'Read Access Specification' = (device object ID, Description)
'Read Access Specification' = (device object ID, Protocol_Version)
'Read Access Specification' = (device object ID, Protocol_Conformance_Class)
'Read Access Specification' = (device object ID, Protocol_Services_Supported)
'Read Access Specification' = (device object ID, Protocol_Object_Types_Supported)
'Read Access Specification' = (device object ID, Max_Message_Length_Supported)
'Read Access Specification' = (device object ID, Window_Size)
'Read Access Specification' = (device object ID, VT_Classes_Supported)
'Read Access Specification' = (device object ID, Active_VT_Sessions)
'Read Access Specification' = (device object ID, Local_Time)
'Read Access Specification' = (device object ID, Local_Date)

IUT:

Process the ReadMultipleProperty request or the sequence of ReadProperty requests and issue a response containing the requested information.

Upper Tester:

No action is required for this test case.

Pass Results:

Option 1 - A Result(+) confirm primitive is received by the lower tester containing a 'List of Read Access Results' argument which contains all of the property values of the device object. These values must match the values specified in the PICS.

Option 2 - A sequence of Result(+) confirm primitives are received by the lower tester, one for each

request issued. Each Result(+) primitive will contain one 'Read Access Result' argument, which conveys the value of one property of the device object. The property values must match the values specified in the PICS.

Fail Results:

This test fails if a Result(-) confirm primitive is received by the lower tester with an error class of OBJECT or an error class of PROPERTY. The test also fails if one or more of the property values returned in the Result(+) primitives is not correct. In this case a configuration error may be the cause and this fact should be noted in the test report.

Inconclusive Results:

The test results are inconclusive if a Result(-) confirm primitive is received by the lower tester with any error class other than OBJECT or PROPERTY. The results are also inconclusive if a reply is not received for one or more of the read requests.

Test A.2.10 Support for the Device Table Object-Type

Test Purpose:

Verify that the implementation supports the Device Table object-type.

BACnet Clause Reference: 5.10

Test Description:

The lower tester issues several ReadProperty requests or a single ReadMultipleProperty request in an attempt to read all of the properties of the Device Table object "Alarm Consoles". The single ReadMultipleProperty request option is preferred if the ReadMultipleProperty service is supported by the IUT.

Expected Result:

A successful attempt to read all of the properties of "Alarm Consoles". The values returned will be the values defined in the standard object configuration.

Lower Tester:

Option 1 - Issue a ReadMultipleProperty service request with the following argument:

'List of Read Access Specifications' = ("Alarm Consoles", ALL)

Option 2 - Issue a sequence of ReadProperty requests, one for each of the four properties of the object. The sequence of arguments is shown below.

'Read Access Specification' = ("Alarm Consoles", Object_Identifier)
'Read Access Specification' = ("Alarm Consoles", Object_Type)
'Read Access Specification' = ("Alarm Consoles", Description)
'Read Access Specification' = ("Alarm Consoles", Device_List)

IUT:

Process the ReadMultipleProperty request or the sequence of ReadProperty requests and issue a response containing the requested information.

Upper Tester:

No action is required for this test case.

Pass Results:

Option 1 - A Result(+) confirm primitive is received by the lower tester containing the following argument:

'List of Read Access Results' = ("Alarm Consoles", DEVICE_TABLE, "Building ABC Host Computer Systems", ("Node 5", "Node 7"))

Option 2 - A sequence of Result(+) confirm primitives are received by the lower tester, one for each request issued. Each Result(+) primitive will contain one argument. The sequence will be:

'Read Access Result' = "Alarm Consoles"
'Read Access Result' = DEVICE_TABLE
'Read Access Result' = "Building ABC Host Computer Systems"
'Read Access Result' = ("Node 5", "Node 7")

Fail Results:

This test fails if a Result(-) confirm primitive is received by the lower tester with an error class of OBJECT or an error class of PROPERTY. The test also fails if one or more of the property values returned in the Result(+) primitives is not correct. In this case a configuration error may be the cause and this fact should be noted in the test report.

Inconclusive Results:

The test results are inconclusive if a Result(-) confirm primitive is received by the lower tester with any error class other than OBJECT or PROPERTY. The results are also inconclusive if a reply is not received for one or more of the read requests.

Test A.2.11 Support for the Directory Object-Type

Test Purpose:

Verify that the implementation supports the Directory object-type.

BACnet Clause Reference: 5.11

Test Description:

The lower tester issues several ReadProperty requests or a single ReadMultipleProperty request in an attempt to read all of the properties of the Directory object "Directory Object 79". The single ReadMultipleProperty request option is preferred if the ReadMultipleProperty service is supported by the IUT.

Expected Result:

A successful attempt to read all of the properties of "Directory Object 79". The values returned will be the values defined in the standard object configuration.

Lower Tester:

Option 1 - Issue a ReadMultipleProperty service request with the following argument:

'List of Read Access Specifications' = ("Directory Object 79", ALL)

Option 2 - Issue a sequence of ReadProperty requests, one for each of the nine properties of the object. The sequence of arguments is shown below.

'Read Access Specification' = ("Directory Object 79", Object_Identifier)
'Read Access Specification' = ("Directory Object 79", Object_Type)
'Read Access Specification' = ("Directory Object 79", Description)
'Read Access Specification' = ("Directory Object 79", Reference_ObjectIdentifier)
'Read Access Specification' = ("Directory Object 79", Reference_ObjectType)
'Read Access Specification' = ("Directory Object 79", Reference_Nickname)
'Read Access Specification' = ("Directory Object 79", Device_Address)
'Read Access Specification' = ("Directory Object 79", Site_ID)
'Read Access Specification' = ("Directory Object 79", Tel_Number)

IUT:

Process the ReadMultipleProperty request or the sequence of ReadProperty requests and issue a response containing the requested information.

Upper Tester:

No action is required for this test case.

Pass Results:

Option 1 - A Result(+) confirm primitive is received by the lower tester containing the following argument:

'List of Read Access Results' = ("Directory Object 79", DIRECTORY,
"Outside Air Temp Obj Dir", OA TEMP-NORTH, ANALOG_INPUT, "OA-MAIN", A5, "North_Shore",
"312-123-4567")

Option 2 - A sequence of Result(+) confirm primitives are received by the lower tester, one for each request issued. Each Result(+) primitive will contain one argument. The sequence will be:

'Read Access Result' = "Directory Object 79"
'Read Access Result' = DIRECTORY
'Read Access Result' = "Outside Air Temp Obj Dir"
'Read Access Result' = "OA TEMP-NORTH"
'Read Access Result' = ANALOG_INPUT
'Read Access Result' = "OA-MAIN"
'Read Access Result' = A5
'Read Access Result' = "North_Shore"
'Read Access Result' = "312-123-4567"

Fail Results:

This test fails if a Result(-) confirm primitive is received by the lower tester with an error class of OBJECT or an error class of PROPERTY. The test also fails if one or more of the property values returned in the Result(+) primitives is not correct. In this case a configuration error may be the cause and this fact should be noted in the test report.

Inconclusive Results:

The test results are inconclusive if a Result(-) confirm primitive is received by the lower tester with any error class other than OBJECT or PROPERTY. The results are also inconclusive if a reply is not received for one or more of the read requests.

Test A.2.12 Support for the Event Enrollment Object-Type

Test Purpose:

Verify that the implementation supports the Event Enrollment object-type.

BACnet Clause Reference: 5.12

Test Description:

The lower tester issues several ReadProperty requests or a single ReadMultipleProperty request in an attempt to read all of the properties of the Event Enrollment object "Zone1ALARM". The single ReadMultipleProperty request option is preferred if the ReadMultipleProperty service is supported by the IUT.

Expected Result:

A successful attempt to read all of the properties of "Zone1ALARM". The values returned will be the values defined in the standard object configuration.

Lower Tester:

Option 1 - Issue a ReadMultipleProperty service request with the following argument:

'List of Read Access Specifications' = ("Zone1ALARM", ALL)

Option 2 - Issue a sequence of ReadProperty requests, one for each of the eleven properties of the object. The sequence of arguments is shown below.

'Read Access Specification' = ("Zone1ALARM", Object_Identifier)
'Read Access Specification' = ("Zone1ALARM", Object_Type)
'Read Access Specification' = ("Zone1ALARM", Event_Type)
'Read Access Specification' = ("Zone1ALARM", Property_Reference)
'Read Access Specification' = ("Zone1ALARM", State)
'Read Access Specification' = ("Zone1ALARM", Enabled)
'Read Access Specification' = ("Zone1ALARM", Acknowledged)
'Read Access Specification' = ("Zone1ALARM", Notification_Rules)
'Read Access Specification' = ("Zone1ALARM", Parameter_List)
'Read Access Specification' = ("Zone1ALARM", Confirmed_Recipient_List)
'Read Access Specification' = ("Zone1ALARM", Unconfirmed_Recipient_List)

IUT:

Process the ReadMultipleProperty request or the sequence of ReadProperty requests and issue a response containing the requested information.

Upper Tester:

No action is required for this test case.

Pass Results:

Option 1 - A Result(+) confirm primitive is received by the lower tester containing the following argument:

'List of Read Access Results' = ("Zone1ALARM", EVENT_ENROLLMENT, ALARM, ("Zone1TEMP", Present_Value), ALARM, TRUE, FALSE, ((NORMAL_TO_ALARM, 2, 3, TRUE, "Alarm"), (ALARM_TO_NORMAL, 5, 3, FALSE, "Normal")), (65., 85., .25), ("Node 29", "Node 96",

"Alarm Consoles"), NULL)

Option 2 - A sequence of Result(+) confirm primitives are received by the lower tester, one for each request issued. Each Result(+) primitive will contain one argument. The sequence will be:

'Read Access Result' = "Zone1ALARM"
'Read Access Result' = EVENT_ENROLLMENT
'Read Access Result' = ALARM
'Read Access Result' = ("Zone1TEMP", Present_Value)
'Read Access Result' = ALARM
'Read Access Result' = TRUE
'Read Access Result' = FALSE
'Read Access Result' = ((NORMAL-TO_ALARM, 2, 3, TRUE, "Alarm"),
(ALARM_TO_NORMAL, 5, 3, FALSE, "Normal"))
'Read Access Result' = 65., 85., .25)
'Read Access Result' = ("Node 29", "Node 96", "Alarm Consoles")
'Read Access Result' = NULL

Fail Results:

This test fails if a Result(-) confirm primitive is received by the lower tester with an error class of OBJECT or an error class of PROPERTY. The test also fails if one or more of the property values returned in the Result(+) primitives is not correct. In this case a configuration error may be the cause and this fact should be noted in the test report.

Inconclusive Results:

The test results are inconclusive if a Result(-) confirm primitive is received by the lower tester with any error class other than OBJECT or PROPERTY. The results are also inconclusive if a reply is not received for one or more of the read requests.

Test A.2.13 Support for the Group Object-Type

Test Purpose:

Verify that the implementation supports the Group object-type.

BACnet Clause Reference: 5.13

Test Description:

The lower tester issues several ReadProperty requests or a single ReadMultipleProperty request in an attempt to read all of the properties of the Group object "Test_Group". The single ReadMultipleProperty request option is preferred if the ReadMultipleProperty service is supported by the IUT.

Expected Result:

A successful attempt to read all of the properties of "Test_Group". The values returned will be the values defined in the standard object configuration.

Lower Tester:

Option 1 - Issue a ReadMultipleProperty service request with the following argument:

'List of Read Access Specifications' = ("Test_Group", ALL)

Option 2 - Issue a sequence of ReadProperty requests, one for each of the four properties of the object.

The sequence of arguments is shown below.

'Read Access Specification' = ("Test_Group", Object_Identifier)
'Read Access Specification' = ("Test_Group", Object_Type)
'Read Access Specification' = ("Test_Group", List_of_Access_Specifications)
'Read Access Specification' = ("Test_Group", Present_Value)

IUT:

Process the ReadMultipleProperty request or the sequence of ReadProperty requests and issue a response containing the requested information.

Upper Tester:

No action is required for this test case.

Pass Results:

Option 1 - A Result(+) confirm primitive is received by the lower tester containing the following argument:

'List of Read Access Results' = ("Test_Group", GROUP, (("1AH1MAT", (Min_Pres, Max_Pres, Description)), ("MA_SetPoint", (Present_Value, Units)), ("Alarm Consoles", (Description, Device_List))), (-50, 250, "Mixed Air Temperature", 58.0, "DEGF", "Building ABC Host Computer Systems", "Node 5", "Node 7"))

Option 2 - A sequence of Result(+) confirm primitives are received by the lower tester, one for each request issued. Each Result(+) primitive will contain one argument. The sequence will be:

'Read Access Result' = "Test_Group"
'Read Access Result' = GROUP
'Read Access Result' = (("1AH1MAT", (Min_Pres, Max_Pres, Description)),
("MA_SetPoint", (Present_Value, Units)),
("Alarm Consoles",(Description,Device_List))),
'Read Access Result' = (-50, 250, "Mixed Air Temperature", 58.0, "DEGF",
"Building ABC Host Computer Systems","Node 5","Node 7"))

Fail Results:

This test fails if a Result(-) confirm primitive is received by the lower tester with an error class of OBJECT or an error class of PROPERTY. The test also fails if one or more of the property values returned in the Result(+) primitives is not correct. In this case a configuration error may be the cause and this fact should be noted in the test report.

Inconclusive Results:

The test results are inconclusive if a Result(-) confirm primitive is received by the lower tester with any error class other than OBJECT or PROPERTY. The results are also inconclusive if a reply is not received for one or more of the read requests.

Test A.2.14 Support for the Loop Object-Type

Test Purpose:

Verify that the implementation supports the Loop object-type.

BACnet Clause Reference: 5.14

Test Description:

The lower tester issues several ReadProperty requests or a single ReadMultipleProperty request in an attempt to read all of the properties of the Loop object "Test_Loop". The single ReadMultipleProperty request option is preferred if the ReadMultipleProperty service is supported by the IUT.

Expected Result:

A successful attempt to read all of the properties of "Test_Loop". The values returned will be the values defined in the standard object configuration.

Lower Tester:

Option 1 - Issue a ReadMultipleProperty service request with the following argument:

'List of Read Access Specifications' = ("Test_Loop", ALL)

Option 2 - Issue a sequence of ReadProperty requests, one for each of the 26 properties of the object. The sequence of arguments is shown below.

'Read Access Specification' = ("Test_Loop", Object_Identifier)
'Read Access Specification' = ("Test_Loop", Object_Type)
'Read Access Specification' = ("Test_Loop", Present_Value)
'Read Access Specification' = ("Test_Loop", Description)
'Read Access Specification' = ("Test_Loop", Status_Flags)
'Read Access Specification' = ("Test_Loop", State)
'Read Access Specification' = ("Test_Loop", Reliability)
'Read Access Specification' = ("Test_Loop", Out_of_Service)
'Read Access Specification' = ("Test_Loop", Update_Interval)
'Read Access Specification' = ("Test_Loop", Output_Units)
'Read Access Specification' = ("Test_Loop", Controlled_Variable_Reference)
'Read Access Specification' = ("Test_Loop", Process_Variable_Reference)
'Read Access Specification' = ("Test_Loop", Process_Variable_Value)
'Read Access Specification' = ("Test_Loop", Process_Units)
'Read Access Specification' = ("Test_Loop", Setpoint_Reference)
'Read Access Specification' = ("Test_Loop", Setpoint)
'Read Access Specification' = ("Test_Loop", Action)
'Read Access Specification' = ("Test_Loop", Proportional_Constant)
'Read Access Specification' = ("Test_Loop", Proportional_Constant_Units)
'Read Access Specification' = ("Test_Loop", Integral_Constant)
'Read Access Specification' = ("Test_Loop", Integral_Constant_Units)
'Read Access Specification' = ("Test_Loop", Derivative_Constant)
'Read Access Specification' = ("Test_Loop", Derivative_Constant_Units)
'Read Access Specification' = ("Test_Loop", Bias)
'Read Access Specification' = ("Test_Loop", Maximum_Output)
'Read Access Specification' = ("Test_Loop", Minimum_Output)

IUT:

Process the ReadMultipleProperty request or the sequence of ReadProperty requests and issue a response containing the requested information.

Upper Tester:

No action is required for this test case.

Pass Results:

Option 1 - A Result(+) confirm primitive is received by the lower tester containing the following argument:

'List of Read Access Results' = ("Test_Loop", LOOP, 8.3, "A simple test loop", {FALSE, TRUE, FALSE, FALSE}, NORMAL, RELIABLE, FALSE, 1, "PSI", ("1AH1DMPR", Present_Value), ("1AH1MAT", Present_Value), 58.1, "DEGF", ("MA_SetPoint", Present_Value), 58, DIRECT, 0.5, "PSI/DEGF", 0.1, "1/MIN", 0, "", 9, 15, 3)

Option 2 - A sequence of Result(+) confirm primitives are received by the lower tester, one for each request issued. Each Result(+) primitive will contain one argument. The sequence will be:

'Read Access Result' = "Test_Loop"
'Read Access Result' = LOOP
'Read Access Result' = 8.3
'Read Access Result' = "A simple test loop"
'Read Access Result' = {FALSE, TRUE, FALSE, FALSE}
'Read Access Result' = NORMAL
'Read Access Result' = RELIABLE
'Read Access Result' = FALSE
'Read Access Result' = 1
'Read Access Result' = "PSI"
'Read Access Result' = ("1AH1DMPR", Present_Value)
'Read Access Result' = ("1AH1MAT", Present_Value)
'Read Access Result' = 58.1
'Read Access Result' = "DEGF"
'Read Access Result' = ("MA_SetPoint", Present_Value)
'Read Access Result' = 58
'Read Access Result' = DIRECT
'Read Access Result' = 0.5
'Read Access Result' = "PSI/DEGF"
'Read Access Result' = 0.1
'Read Access Result' = "1/MIN"
'Read Access Result' = 0
'Read Access Result' = ""
'Read Access Result' = 9
'Read Access Result' = 15
'Read Access Result' = 3

Fail Results:

This test fails if a Result(-) confirm primitive is received by the lower tester with an error class of OBJECT or an error class of PROPERTY. The test also fails if one or more of the property values returned in the Result(+) primitives is not correct. In this case a configuration error may be the cause and this fact should be noted in the test report.

Inconclusive Results:

The test results are inconclusive if a Result(-) confirm primitive is received by the lower tester with any error class other than OBJECT or PROPERTY. The results are also inconclusive if a reply is not received for one or more of the read requests.

Test A.2.15 Support for the Multi-State Input Object-Type

Test Purpose:

Verify that the implementation supports the Multi-State Input object-type.

BACnet Clause Reference: 5.15

Test Description:

The lower tester issues several ReadProperty requests or a single ReadMultipleProperty request in an attempt to read all of the properties of the Multi-State Input object "Fan1_Input". The single ReadMultipleProperty request option is preferred if the ReadMultipleProperty service is supported by the IUT.

Expected Result:

A successful attempt to read all of the properties of "Fan1_Input". The values returned will be the values defined in the standard object configuration.

Lower Tester:

Option 1 - Issue a ReadMultipleProperty service request with the following argument:

'List of Read Access Specifications' = ("Fan1_Input", ALL)

Option 2 - Issue a sequence of ReadProperty requests, one for each of the ten properties of the object. The sequence of arguments is shown below.

'Read Access Specification' = ("Fan1_Input", Object_Identifier)
'Read Access Specification' = ("Fan1_Input", Object_Type)
'Read Access Specification' = ("Fan1_Input", Present_Value)
'Read Access Specification' = ("Fan1_Input", Description)
'Read Access Specification' = ("Fan1_Input", Status_Flags)
'Read Access Specification' = ("Fan1_Input", State)
'Read Access Specification' = ("Fan1_Input", Reliability)
'Read Access Specification' = ("Fan1_Input", Out_of_Service)
'Read Access Specification' = ("Fan1_Input", Number_of_States)
'Read Access Specification' = ("Fan1_Input", State_Text)

IUT:

Process the ReadMultipleProperty request or the sequence of ReadProperty requests and issue a response containing the requested information.

Upper Tester:

No action is required for this test case.

Pass Results:

Option 1 - A Result(+) confirm primitive is received by the lower tester containing the following argument:

'List of Read Access Results' = ("Fan1_Input", MULTI-STATE_INPUT, STATE_2, "2-speed Fan#1", {FALSE, TRUE, FALSE, FALSE}, NORMAL, RELIABLE, FALSE, 3, ("Off", "On_Low", "On-High"))

Option 2 - A sequence of Result(+) confirm primitives are received by the lower tester, one for each

request issued. Each Result(+) primitive will contain one argument. The sequence will be:

'Read Access Result' = "Fan1_Input"
'Read Access Result' = MULTI-STATE_INPUT
'Read Access Result' = STATE_2
'Read Access Result' = "2-speed Fan#1"
'Read Access Result' = {FALSE, TRUE, FALSE, FALSE}
'Read Access Result' = NORMAL
'Read Access Result' = RELIABLE
'Read Access Result' = FALSE
'Read Access Result' = 3
'Read Access Result' = ("Off", "On_Low", "On_High")

Fail Results:

This test fails if a Result(-) confirm primitive is received by the lower tester with an error class of OBJECT or an error class of PROPERTY. The test also fails if one or more of the property values returned in the Result(+) primitives is not correct. In this case a configuration error may be the cause and this fact should be noted in the test report.

Inconclusive Results:

The test results are inconclusive if a Result(-) confirm primitive is received by the lower tester with any error class other than OBJECT or PROPERTY. The results are also inconclusive if a reply is not received for one or more of the read requests.

Test A.2.16 Support for the Multi-State Output Object-Type

Test Purpose:

Verify that the implementation supports the Multi-State Output object-type.

BACnet Clause Reference: 5.16

Test Description:

The lower tester issues several ReadProperty requests or a single ReadMultipleProperty request in an attempt to read all of the properties of the Multi-State Output object "Fan1_Output". The single ReadMultipleProperty request option is preferred if the ReadMultipleProperty service is supported by the IUT.

Expected Result:

A successful attempt to read all of the properties of "Fan1_Output". The values returned will be the values defined in the standard object configuration.

Lower Tester:

Option 1 - Issue a ReadMultipleProperty service request with the following argument:

'List of Read Access Specifications' = ("Fan1_Output", ALL)

Option 2 - Issue a sequence of ReadProperty requests, one for each of the ten properties of the object. The sequence of arguments is shown below.

'Read Access Specification' = ("Fan1_Output", Object_Identifier)

'Read Access Specification' = ("Fan1_Output", Object_Type)
'Read Access Specification' = ("Fan1_Output", Present_Value)
'Read Access Specification' = ("Fan1_Output", Description)
'Read Access Specification' = ("Fan1_Output", Status_Flags)
'Read Access Specification' = ("Fan1_Output", State)
'Read Access Specification' = ("Fan1_Output", Reliability)
'Read Access Specification' = ("Fan1_Output", Out_of_Service)
'Read Access Specification' = ("Fan1_Output", Number_of_States)
'Read Access Specification' = ("Fan1_Output", State_Text)

IUT:

Process the ReadMultipleProperty request or the sequence of ReadProperty requests and issue a response containing the requested information.

Upper Tester:

No action is required for this test case.

Pass Results:

Option 1 - A Result(+) confirm primitive is received by the lower tester containing the following argument:

'List of Read Access Results' = ("Fan1_Output", MULTI-STATE_OUTPUT, STATE_2, "2-speed Fan#1", {FALSE, TRUE, FALSE, FALSE}, NORMAL, RELIABLE, FALSE, 3, ("Off", "On_Low", "On-High","ON-LOW")

Option 2 - A sequence of Result(+) confirm primitives are received by the lower tester, one for each request issued. Each Result(+) primitive will contain one argument. The sequence will be:

'Read Access Result' = "Fan1_Output"
'Read Access Result' = MULTI-STATE_OUTPUT
'Read Access Result' = STATE_2
'Read Access Result' = "2-speed Fan#1"
'Read Access Result' = {FALSE, TRUE, FALSE, FALSE}
'Read Access Result' = NORMAL
'Read Access Result' = RELIABLE
'Read Access Result' = FALSE
'Read Access Result' = 3
'Read Access Result' = ("Off", "On_Low", On_High")

Fail Results:

This test fails if a Result(-) confirm primitive is received by the lower tester with an error class of OBJECT or an error class of PROPERTY. The test also fails if one or more of the property values returned in the Result(+) primitives is not correct. In this case a configuration error may be the cause and this fact should be noted in the test report.

Inconclusive Results:

The test results are inconclusive if a Result(-) confirm primitive is received by the lower tester with any error class other than OBJECT or PROPERTY. The results are also inconclusive if a reply is not received for one or more of the read requests.

Test A.2.17 Support for the Program Object-Type

Test Purpose:

Verify that the implementation supports the Program object-type.

BACnet Clause Reference: 5.17

Test Description:

The lower tester issues several ReadProperty requests or a single ReadMultipleProperty request in an attempt to read all of the properties of the program object specified in the PICS. The single ReadMultipleProperty request option is preferred if the ReadMultipleProperty service is supported by the IUT.

Expected Result:

A successful attempt to read all of the properties of the program object. Since the program object is vendor-specific, the values returned will be the values defined in the PICS.

Lower Tester:

Option 1 - Issue a ReadMultipleProperty service request with the following argument:

'List of Read Access Specifications' = (program object ID, ALL)

Option 2 - Issue a sequence of ReadProperty requests, one for each of the eleven properties of the object. The sequence of arguments is shown below.

'Read Access Specification' = (program object ID, Object_Identifier)
'Read Access Specification' = (program object ID, Object_Type)
'Read Access Specification' = (program object ID, Program_State)
'Read Access Specification' = (program object ID, Program_Change)
'Read Access Specification' = (program object ID, Reason_For_Halt)
'Read Access Specification' = (program object ID, Description)
'Read Access Specification' = (program object ID, Instance_Of)
'Read Access Specification' = (program object ID, Status_Flags)
'Read Access Specification' = (program object ID, State)
'Read Access Specification' = (program object ID, Reliability
'Read Access Specification' = (program object ID, Out_Of_Service)

IUT:

Process the ReadMultipleProperty request or the sequence of ReadProperty requests and issue a response containing the requested information.

Upper Tester:

No action is required for this test case.

Pass Results:

Option 1 - A Result(+) confirm primitive is received by the lower tester containing a 'List of Read Access Results' argument which contains all of the property values of the program object. These values must match the values specified in the PICS.

Option 2 - A sequence of Result(+) confirm primitives are received by the lower tester, one for each request issued. Each Result(+) primitive will contain one 'Read Access Result' argument, which conveys

the value of one property of the program object. The property values must match the values specified in the PICS.

Fail Results:

This test fails if a Result(-) confirm primitive is received by the lower tester with an error class of OBJECT or an error class of PROPERTY. The test also fails if one or more of the property values returned in the Result(+) primitives is not correct. In this case a configuration error may be the cause and this fact should be noted in the test report.

Inconclusive Results:

The test results are inconclusive if a Result(-) confirm primitive is received by the lower tester with any error class other than OBJECT or PROPERTY. The results are also inconclusive if a reply is not received for one or more of the read requests.

Test A.2.18 Support for the Schedule Object-Type

Test Purpose:

Verify that the implementation supports the Schedule object-type.

BACnet Clause Reference: 5.18

Test Description:

The lower tester issues several ReadProperty requests or a single ReadMultipleProperty request in an attempt to read all of the properties of the Schedule object "Rm208Sched". The single ReadMultipleProperty request option is preferred if the ReadMultipleProperty service is supported by the IUT.

Expected Result:

A successful attempt to read all of the properties of "Rm208Sched". The values returned will be the values defined in the standard object configuration.

Lower Tester:

Option 1 - Issue a ReadMultipleProperty service request with the following argument:

'List of Read Access Specifications' = ("Rm208Sched", ALL)

Option 2 - Issue a sequence of ReadProperty requests, one for each of the fourteen properties of the object. The sequence of arguments is shown below.

'Read Access Specification' = ("Rm208Sched", Object_Identifier)
'Read Access Specification' = ("Rm208Sched", Object_Type)
'Read Access Specification' = ("Rm208Sched", Present_Value)
'Read Access Specification' = ("Rm208Sched", Effective_Period)
'Read Access Specification' = ("Rm208Sched", Monday_Schedule)
'Read Access Specification' = ("Rm208Sched", Tuesday_Schedule)
'Read Access Specification' = ("Rm208Sched", Wednesday_Schedule)
'Read Access Specification' = ("Rm208Sched", Thursday_Schedule)
'Read Access Specification' = ("Rm208Sched", Friday_Schedule)
'Read Access Specification' = ("Rm208Sched", Saturday_Schedule)
'Read Access Specification' = ("Rm208Sched", Sunday_Schedule)

'Read Access Specification' = ("Rm208Sched", Exception_Schedule)
'Read Access Specification' = ("Rm208Sched", Property_Reference)
'Read Access Specification' = ("Rm208Sched", Priority_For_Writing)

IUT:

Process the ReadMultipleProperty request or the sequence of ReadProperty requests and issue a response containing the requested information.

Upper Tester:

No action is required for this test case.

Pass Results:

Option 1 - A Result(+) confirm primitive is received by the lower tester containing the following argument:

'List of Read Access Results' = ("Rm208Sched", SCHEDULE, ACTIVE, ((90/9/5), (91/6/10)), ((8:00, ACTIVE), (17:00, INACTIVE)), ((8:00, ACTIVE)), ((8:00, ACTIVE), (17:00, INACTIVE)), ((8:00 ACTIVE), (17:00, INACTIVE), (19:00, ACTIVE), (23:30, INACTIVE)), ((8:00, ACTIVE), (17:00, INACTIVE)), ((10:00, ACTIVE), (17:00, INACTIVE)), ((0:00, INACTIVE)), (((90/11/23), (0:00, INACTIVE), 10), (HOLIDAYS, (0:00, INACTIVE), 11),(((91/3/5), (91/3/7)), ((9:00, ACTIVE), (14:00, INACTIVE)),6)))

Option 2 - A sequence of Result(+) confirm primitives are received by the lower tester, one for each request issued. Each Result(+) primitive will contain one argument. The sequence will be:

'Read Access Result' = "Rm208Sched"
'Read Access Result' = SCHEDULE
'Read Access Result' = ACTIVE
'Read Access Result' = ((90/9/5), (91/6/10))
'Read Access Result' = ((8:00, ACTIVE), (17:00, INACTIVE))
'Read Access Result' = ((8:00, ACTIVE))
'Read Access Result' = ((8:00, ACTIVE), (17:00, INACTIVE))
'Read Access Result' = ((8:00, ACTIVE), (17:00, INACTIVE),
(19:00, ACTIVE), 23:30, INACTIVE))
'Read Access Result' = ((8:00, ACTIVE), (17:00, INACTIVE))
'Read Access Result' = ((10:00, ACTIVE), (17:00, INACTIVE))
'Read Access Result' = ((0:00, INACTIVE))
'Read Access Result' = (((90/11/23), (0:00, INACTIVE), 10),
(HOLIDAYS, (0:00, INACTIVE), 11),
(((91/3/5), (91/3/7)), ((9:00, ACTIVE), (14:00, INACTIVE)),6))

Fail Results:

This test fails if a Result(-) confirm primitive is received by the lower tester with an error class of OBJECT or an error class of PROPERTY. The test also fails if one or more of the property values returned in the Result(+) primitives is not correct. In this case a configuration error may be the cause and this fact should be noted in the test report.

Inconclusive Results:

The test results are inconclusive if a Result(-) confirm primitive is received by the lower tester with any error class other than OBJECT or PROPERTY. The results are also inconclusive if a reply is not received for one or more of the read requests.

6.2 Property Functional Range Tests

Functional range tests are used to verify that an implementation will support property values over the entire range specified by BACnet. This is done by writing selected values to the properties that span the allowable range and subsequently reading the values back to verify that the write attempt was successful. It is assumed in this abstract test suite that if a property is correctly supported for one object-type that any other object-type that has the same property is also correct. Thus, each property is tested only once.

The current draft of BACnet specifies 100 different properties. For many of these properties a finite range of values has not yet been specified so functional range testing cannot be done. A common example is a property with a datatype of Unsigned. In theory this is an integer with a range from zero to infinity. There are proposals to define integer datatypes with more limited ranges to in order to manage storage requirements but a final decision in this mater has not yet been made. Many of the properties defined in BACnet are character strings. This is another kind of property which does not have a fixed bound. This may also need to be changed because of a need to manage storage in a device. There are properties that are intimately linked to application services and their functional range can only be tested in the context of testing the application service. The Acknowledged property of Event Enrollment objects is an example of such a property. Its value is set to FALSE when an alarm occurs and is set to TRUE when an AcknowledgeAlarm indication is received. The remaining properties are enumerated values or have an explicitly defined ranges. Functional range tests for these properties can be defined at this time. Appendix 1 is an abstract syntax notation one (ASN.1) description of the BACnet property identifiers. Comments indicate which of the categories above applies to each of the properties.

Test A.3.1 Functional Range for the Action Property

Test Purpose:

Verify that the IUT supports values for the Action property over the entire defined range.

BACnet Clause Reference: 5.14.1.17

Test Description:

The lower tester verifies the initial value of the property then alternately writes and reads new values of the property until the entire range of values is spanned.

Expected Results:

The Action property has only two defined values, DIRECT and REVERSE. The lower tester should be able to write and then read either or both of these values.

Lower Tester:

The lower tester takes the following steps.

1. Issue a ReadProperty service request with a 'Read Access Specificaiton' = ("Test_Loop", Action).

2. Issue a WriteProperty service request with a 'Write Access Specification' = (NULL, "Test_Loop", Action, REVERSE).

3. Issue a ReadProperty service request with a 'Read Access Specification' = ("Test_Loop", Action).

4. Issue a WriteProperty service request with a 'Write Access Specification' = (NULL, "Test_Loop", Action, DIRECT).

IUT:

Process the ReadProperty and WriteProperty indications as specified by BACnet.

Upper Tester:

No action is required for this test.

Pass Result:

The lower tester receives a confirm primitive corresponding to each of the four service requests described above. The responses in order are:

1. A Result(+) confirm primitive conveying the parameter 'Read Access Result' = ("Test_Loop", Action, DIRECT).

2. A Result(+) confirm primitive.

3. A Result(+) confirm primitive conveying the parameter 'Read Access Result' = ("Test_Loop", Action, REVERSE).

4. A Result(+) confirm primitive.

Fail Result:

If the WriteProperty service is successfully completed and the value for the Action property returned in step 1 or step 3 is not as specified then the result is failure. If a Result(-) confirm primitive is received in response to any of the service requests and the error class is PROPERTY then the test result is failure.

Inconclusive Result:

If the IUT does not respond or if a Result(-) confirm primitive is received in response to any of the service requests and the error class is *not* PROPERTY the result is inconclusive.

Test A.3.2 Functional Range for the Enabled Property

Test Purpose:

Verify that the IUT supports values for the Enabled property over the entire defined range.

BACnet Clause Reference: 5.14.1.17

Test Description:

The lower tester verifies the initial value of the property then alternately writes and reads new values of the property until the entire range of values is spanned.

Expected Results:

The Enabled property has only two defined values, TRUE and FALSE. The lower tester should be able to write and then read either or both of these values.

Lower Tester:

The lower tester takes the following steps.

1. Issue a ReadProperty service request with a 'Read Access Specificaiton' = ("Zone1ALARM", Enabled).

2. Issue a WriteProperty service request with a 'Write Access Specification' = (NULL, "Zone1ALARM", Enabled, FALSE).

3. Issue a ReadProperty service request with a 'Read Access Specification' = ("Zone1ALARM", Enabled).

4. Issue a WriteProperty service request with a 'Write Access Specification' = (NULL, "Zone1ALARM", Enabled, TRUE).

IUT:

Process the ReadProperty and WriteProperty indications as specified by BACnet.

Upper Tester:

No action is required for this test.

Pass Result:

The lower tester receives a confirm primitive corresponding to each of the four service requests described above. The responses in order are:

1. A Result(+) confirm primitive conveying the parameter 'Read Access Result' = ("Zone1ALARM", Enabled, TRUE).

2. A Result(+) confirm primitive.

3. A Result(+) confirm primitive conveying the parameter 'Read Access Result' = ("Zone1ALARM", Enabled, FALSE).

4. A Result(+) confirm primitive.

Fail Result:

If the WriteProperty service is successfully completed and the value for the Enabled property returned in step 1 or step 3 is not as specified then the result is failure. If a Result(-) confirm primitive is received in response to any of the service requests and the error class is PROPERTY then the test result is failure.

Inconclusive Result:

If the IUT does not respond or if a Result(-) confirm primitive is received in response to any of the service requests and the error class is *not* PROPERTY the result is inconclusive.

Test A.3.3 Functional Range for the Event_Type Property

Test Purpose:

Verify that the IUT supports values for the Event_Type property over the entire defined range.

BACnet Clause Reference: 5.14.1.17

Test Description:

The lower tester verifies the initial value of the property then alternately writes and reads new values of the property until the entire range of values is spanned.

Expected Results:

The Event_Type property has seven defined values (CHANGE_OF_VALUE, OUT_OF_RANGE, CHANGE_OF_STATE_EVENT, CHANGE_OF_STATE_ALARM, OUT_OF_RANGE_ALARM, FLOATING_LIMIT_ALARM, COMMAND_FAIL_ALARM). The lower tester should be able to write and then read any of these values.

Lower Tester:

The lower tester takes the following steps.

1. Issue a ReadProperty service request with a 'Read Access Specificaiton' = ("Zone1ALARM", Event_Type).

2. Issue a WriteProperty service request with a 'Write Access Specification' = (NULL, "Zone1ALARM", Event_Type, CHANGE_OF_VALUE).

3. Issue a ReadProperty service request with a 'Read Access Specification' = ("Zone1ALARM", Event_Type).

4. Issue a WriteProperty service request with a 'Write Access Specification' = (NULL, "Zone1ALARM", Event_Type, OUT_OF_RANGE).

5. Issue a ReadProperty service request with a 'Read Access Specificaiton' = ("Zone1ALARM", Event_Type).

6. Issue a WriteProperty service request with a 'Write Access Specification' = (NULL, "Zone1ALARM", Event_Type, CHANGE_OF_STATE_EVENT).

7. Issue a ReadProperty service request with a 'Read Access Specification' = ("Zone1ALARM", Event_Type).

8. Issue a WriteProperty service request with a 'Write Access Specification' = (NULL, "Zone1ALARM", Event_Type, CHANGE_OF_STATE_ALARM).

9. Issue a ReadProperty service request with a 'Read Access Specificaiton' = ("Zone1ALARM", Event_Type).

10. Issue a WriteProperty service request with a 'Write Access Specification' = (NULL, "Zone1ALARM", Event_Type, FLOATING_LIMIT_ALARM).

11. Issue a ReadProperty service request with a 'Read Access Specification' = ("Zone1ALARM", Event_Type).

12. Issue a WriteProperty service request with a 'Write Access Specification' = (NULL, "Zone1ALARM", Event_Type, COMMAND_FAIL_ALARM).

13. Issue a ReadProperty service request with a 'Read Access Specificaiton' = ("Zone1ALARM", Event_Type).

14. Issue a WriteProperty service request with a 'Write Access Specification' = (NULL, "Zone1ALARM", Event_Type, OUT_OF_RANGE_ALARM).

IUT:

Process the ReadProperty and WriteProperty indications as specified by BACnet.

Upper Tester:

No action is required for this test.

Pass Result:

The lower tester receives a confirm primitive corresponding to each of the four service requests described above. The responses in order are:

1. A Result(+) confirm primitive conveying the parameter 'Read Access Result' = ("Zone1ALARM", Event_Type, OUT_OF_RANGE_ALARM).

2. A Result(+) confirm primitive.

3. A Result(+) confirm primitive conveying the parameter 'Read Access Result' = ("Zone1ALARM", Event_Type, CHANGE_OF_VALUE).

4. A Result(+) confirm primitive.

5. A Result(+) confirm primitive conveying the parameter 'Read Access Result' = ("Zone1ALARM", Event_Type, OUT_OF_RANGE).

6. A Result(+) confirm primitive.

7. A Result(+) confirm primitive conveying the parameter 'Read Access Result' = ("Zone1ALARM", Event_Type, CHANGE_OF_STATE_EVENT).

8. A Result(+) confirm primitive.

9. A Result(+) confirm primitive conveying the parameter 'Read Access Result' = ("Zone1ALARM", Event_Type, CHANGE_OF_STATE_ALARM).

10. A Result(+) confirm primitive.

11. A Result(+) confirm primitive conveying the parameter 'Read Access Result' = ("Zone1ALARM", Event_Type, FLOATING_LIMIT_ALARM).

12. A Result(+) confirm primitive.

13. A Result(+) confirm primitive conveying the parameter 'Read Access Result' = ("Zone1ALARM", Event_Type, COMMAND_FAIL_ALARM).

14. A Result(+) confirm primitive.

Fail Result:

If the WriteProperty service is successfully completed and the value for the Event_Type property returned in steps 1, 3, 5, 7, 9, 11, or 13 is not as specified then the result is failure. If a Result(-) confirm primitive is received in response to any of the service requests and the error class is PROPERTY then the test result is failure.

Inconclusive Result:

If the IUT does not respond or if a Result(-) confirm primitive is received in response to any of the service requests and the error class is not PROPERTY, the result is inconclusive.

Test A.3.4 Functional Range for the Minimum_Off_Time Property

Test Purpose:

Verify that the IUT supports values for the Minimun_Off_Time property over the entire defined range.

note: This version of the test assumes a datatype of Unsigned16 instead of the specified Unsigned32. This is an upper bound of 18 hours instead of 136 years.

BACnet Clause Reference: 5.5.1.16

Test Description:

The lower tester verifies the initial value of the property then alternately writes and reads new values of the property until the entire range of values is spanned.

Expected Results:

The Minimum_Off_Time property has a range from 0 - 65535 seconds. The lower tester should be able to write and then read any value in this range.

Lower Tester:

The lower tester takes the following steps.

1. Issue a WriteProperty service request with a 'Write Access Specification' = (NULL, "Floor3ExhaustFan", Minimum_Off_Time, 0).

2. Issue a ReadProperty service request with a 'Read Access Specification' = ("Floor3ExhaustFan", Minimum_Off_Time).

3. Issue a WriteProperty service request with a 'Write Access Specification' = (NULL, "Floor3ExhaustFan", Minimum_Off_Time, 65535).

4. Issue a ReadProperty service request with a 'Read Access Specification' = ("Floor3ExhaustFan", Minimum_Off_Time).

5. Issue a WriteProperty service request with a 'Write Access Specification' = (NULL, "Floor3ExhaustFan", Minimum_Off_Time, 100).

IUT:

Process the ReadProperty and WriteProperty indications as specified by BACnet.

Upper Tester:

No action is required for this test.

Pass Result:

The lower tester receives a confirm primitive corresponding to each of the four service requests described above. The responses in order are:

1. A Result(+) confirm primitive.

2. A Result(+) confirm primitive conveying the parameter 'Read Access Result' = ("Floor3ExhaustFan", Minimum_Off_Time, 0).

3. A Result(+) confirm primitive.

4. A Result(+) confirm primitive conveying the parameter 'Read Access Result' = ("Floor3ExhaustFan", Minimum_Off_Time, 65535).

5. A Result(+) confirm primitive.

Fail Result:

If the WriteProperty service is successfully completed and the value for the Minimum_Off_Time property returned in step 1 or step 3 is not as specified then the result is failure. If a Result(-) confirm primitive is received in response to any of the service requests and the error class is PROPERTY, then the test result is failure.

Inconclusive Result:

If the IUT does not respond or if a Result(-) confirm primitive is received in response to any of the service requests and the error class is not PROPERTY, the result is inconclusive.

Test A.3.5 Functional Range for the Minimum_On_Time Property

Test Purpose:

Verify that the IUT supports values for the Minimun_On_Time property over the entire defined range.

note: This version of the test assumes a datatype of Unsigned16 instead of the specified Unsigned32. This is an upper bound of 18 hours instead of 136 years.

BACnet Clause Reference: 5.5.1.17

Test Description:

The lower tester verifies the initial value of the property then alternately writes and reads new values of the property until the entire range of values is spanned.

Expected Results:

The Minimum_On_Time property has a range from 0 - 65535 seconds. The lower tester should be able to write and then read any value in this range.

Lower Tester:

The lower tester takes the following steps.

1. Issue a WriteProperty service request with a 'Write Access Specification' = (NULL, "Floor3ExhaustFan", Minimum_On_Time, 0).

2. Issue a ReadProperty service request with a 'Read Access Specification' = ("Floor3ExhaustFan", Minimum_On_Time).

3. Issue a WriteProperty service request with a 'Write Access Specification' = (NULL, "Floor3ExhaustFan", Minimum_On_Time, 65535).

4. Issue a ReadProperty service request with a 'Read Access Specification' = ("Floor3ExhaustFan", Minimum_On_Time).

5. Issue a WriteProperty service request with a 'Write Access Specification' = (NULL, "Floor3ExhaustFan", Minimum_On_Time, 10).

IUT:

Process the ReadProperty and WriteProperty indications as specified by BACnet.

Upper Tester:

No action is required for this test.

Pass Result:

The lower tester receives a confirm primitive corresponding to each of the four service requests described above. The responses in order are:

1. A Result(+) confirm primitive.

2. A Result(+) confirm primitive conveying the parameter 'Read Access Result' = ("Floor3ExhaustFan", Minimum_On_Time, 0).

3. A Result(+) confirm primitive.

4. A Result(+) confirm primitive conveying the parameter 'Read Access Result' = ("Floor3ExhaustFan", Minimum_On_Time, 65535).

5. A Result(+) confirm primitive.

Fail Result:

If the WriteProperty service is successfully completed and the value for the Minimum_On_Time property returned in step 1 or step 3 is not as specified then the result is failure. If a Result(-) confirm primitive is received in response to any of the service requests and the error class is PROPERTY, then the test result is failure.

Inconclusive Result:

If the IUT does not respond or if a Result(-) confirm primitive is received in response to any of the service requests and the error class is not PROPERTY, the result is inconclusive.

Test A.3.6 Functional Range for the Out_Of_Service Property

Test Purpose:

Verify that the IUT supports values for the Out_Of_Service property over the entire defined range.

BACnet Clause Reference: 5.1.1.9

Test Description:

The lower tester verifies the initial value of the property then alternately writes and reads new values of the property until the entire range of values is spanned.

Expected Results:

The Out_Of_Service property has only two defined values, TRUE and FALSE. The lower tester should be able to write and then read either or both of these values.

Lower Tester:

The lower tester takes the following steps.

1. Issue a ReadProperty service request with a 'Read Access Specificaiton' = ("1AH1MAT", Out_Of_Service).

2. Issue a WriteProperty service request with a 'Write Access Specification' = (NULL, "1AH1MAT", Out_Of_Service, TRUE).

3. Issue a ReadProperty service request with a 'Read Access Specification' = ("1AH1MAT", Out_Of_Service).

4. Issue a WriteProperty service request with a 'Write Access Specification' = (NULL, "1AH1MAT", Out_Of_Service, FALSE).

IUT:

Process the ReadProperty and WriteProperty indications as specified by BACnet.

Upper Tester:

No action is required for this test.

Pass Result:

The lower tester receives a confirm primitive corresponding to each of the four service requests described above. The responses in order are:

1. A Result(+) confirm primitive conveying the parameter 'Read Access Result' = ("1AH1MAT", Out_Of_Service, FALSE).

2. A Result(+) confirm primitive.

3. A Result(+) confirm primitive conveying the parameter 'Read Access Result' = ("1AH1MAT", Out_Of_Service, TRUE)

4. A Result(+) confirm primitive.

Fail Result:

If the WriteProperty service is successfully completed and the value for the Out_Of_Service property returned in step 1 or step 3 is not as specified then the result is failure. If a Result(-) confirm primitive is received in response to any of the service requests and the error class is PROPERTY, then the test result is failure.

Inconclusive Result:

If the IUT does not respond or if a Result(-) confirm primitive is received in response to any of the service requests and the error class is not PROPERTY, the result is inconclusive.

Test A.3.7 Functional Range for the Polarity Property

Test Purpose:

Verify that the IUT supports values for the POLARITY property over the entire defined range.

BACnet Clause Reference: 5.4.1.9

Test Description:

The lower tester verifies the initial value of the property then alternately writes and reads new values of the property until the entire range of values is spanned.

Expected Results:

The Polarity property has only two defined values, NORMAL and REVERSE. The lower tester should be able to write and then read either or both of these values.

Lower Tester:

The lower tester takes the following steps.

1. Issue a ReadProperty service request with a 'Read Access Specificaiton' = ("HighPressSwitch", Polarity).

2. Issue a WriteProperty service request with a 'Write Access Specification' = (NULL, "HighPressSwitch", Polarity, REVERSE).

3. Issue a ReadProperty service request with a 'Read Access Specification' = ("HighPressSwitch", Polarity).

4. Issue a WriteProperty service request with a 'Write Access Specification' = (NULL, "HighPressSwitch", Polarity, NORMAL).

IUT:

Process the ReadProperty and WriteProperty indications as specified by BACnet.

Upper Tester:

No action is required for this test.

Pass Result:

The lower tester receives a confirm primitive corresponding to each of the four service requests described above. The responses in order are:

1. A Result(+) confirm primitive conveying the parameter 'Read Access Result' = ("HighPressSwitch", Polarity, NORMAL).

2. A Result(+) confirm primitive.

3. A Result(+) confirm primitive conveying the parameter 'Read Access Result' = ("HighPressSwitch", Polarity, REVERSE).

4. A Result(+) confirm primitive.

Fail Result:

If the WriteProperty service is successfully completed and the value for the Polarity property returned in step 1 or step 3 is not as specified then the result is failure. If a Result(-) confirm primitive is received in response to any of the service requests and the error class is PROPERTY, then the test result is failure.

Inconclusive Result:

If the IUT does not respond or if a Result(-) confirm primitive is received in response to any of the service requests and the error class is not PROPERTY, the result is inconclusive.

Test A.3.8 Functional Range for the Run_Time Property

Test Purpose:

Verify that the IUT supports values for the Run_Time property over the entire defined range.

BACnet Clause Reference: 5.5.1.13

Test Description:

The lower tester verifies the initial value of the property then alternately writes and reads new values of the property until the entire range of values is spanned.

Expected Results:

The Run_Time property has a range from 0 - (2^{32}-1) seconds. The lower tester should be able to write and then read any value in this range.

Lower Tester:

The lower tester takes the following steps.

1. Issue a WriteProperty service request with a 'Write Access Specification' = (NULL, "Floor3ExhaustFan", Run_Time, 0).

2. Issue a ReadProperty service request with a 'Read Access Specification' = ("Floor3ExhaustFan", Run_Time).

3. Issue a WriteProperty service request with a 'Write Access Specification' = (NULL, "Floor3ExhaustFan", Run_Time, 2^{32}-1).

4. Issue a ReadProperty service request with a 'Read Access Specification' = ("Floor3ExhaustFan", Run_Time).

5. Issue a WriteProperty service request with a 'Write Access Specification' = (NULL, "Floor3ExhaustFan", Run_Time, 401).

IUT:

Process the ReadProperty and WriteProperty indications as specified by BACnet.

Upper Tester:

No action is required for this test.

Pass Result:

The lower tester receives a confirm primitive corresponding to each of the four service requests described above. The responses in order are:

1. A Result(+) confirm primitive.

2. A Result(+) confirm primitive conveying the parameter 'Read Access Result' = ("Floor3ExhaustFan", Minimum_On_Time, 0).

3. A Result(+) confirm primitive.

4. A Result(+) confirm primitive conveying the parameter 'Read Access Result' = ("Floor3ExhaustFan",

Minimum_On_Time, 2^{32}-1).

5. A Result(+) confirm primitive.

Fail Result:

If the WriteProperty service is successfully completed and the value for the Run_Time property returned in step 1 or step 3 is not as specified then the result is failure. If a Result(-) confirm primitive is received in response to any of the service requests and the error class is PROPERTY, then the test result is failure.

Inconclusive Result:

If the IUT does not respond or if a Result(-) confirm primitive is received in response to any of the service requests and the error class is not PROPERTY, the result is inconclusive.

7. BEHAVIOR TESTS

The current draft of BACnet defines 29 application services. The behavior tests check the conformance of all of these services. The IUT is tested for its response to both valid and invalid requests from the lower tester and the IUT is instructed to generate application service requests of its own to test this capability.

7.1 Responses to Valid Behavior by a Peer Implementation

All test cases in this group are based on valid application service requests. They are divided into a hierarchy consisting of two main groups, each of which is further broken down into five smaller groups representing the five classes of services defined in BACnet. The two main groups are services initiated by the lower tester and services initiated by the IUT. The services initiated by the lower tester are designed to test the IUT's ability to correctly respond to valid service requests. The services initiated by the IUT are designed to test the IUT's ability to correctly generate valid application service requests.

Some tests require that the properties of the various objects in the IUT have particular values which are not the standard configuration. For these tests there are one or more WriteProperty service requests issued to correctly establish the configuration. At the completion of the test the object database is returned to the standard configuration so that the IUT will be in a known state for beginning the next test in the suite.

7.1.1 Application Services Initiated by the Lower Tester

For all tests in this group the lower tester issues a correctly formed application service request and the response from the IUT is analyzed to determine if it correctly responds to the request as specified in the service procedure.

7.1.1.1 Alarm and Event Services

Test B.1.1.1 Support for the AcknowledgeAlarm Service

Test Purpose:

Verify that the IUT can correctly respond to a valid AcknowledgeAlarm service request. This test also serves to verify the functional range of the Acknowledged property.

BACnet Clause Reference: 5.12.1.7, 6.3.1

Test Description:

The standard object configuration contains an Event Enrollment object with an unacknowledged alarm. The lower tester acknowledges this alarm and then reads the Acknowledged property to ensure that it has been reset.

Expected Result:

A confirmation that the AcknowledgeAlarm service indication was received and the Acknowledged property reset to TRUE.

Lower Tester:

The lower tester issues an AcknowledgeAlarm service request conveying the following parameters.

'Event Enrollment Name'	= "Zone1Alarm"
'Acknowledged State'	= ALARM
'Time Stamp'	= current time

After the request is issued several things can happen and the lower tester must respond as indicated below.

IUT Response PDU	Lower Tester Action
none	When a timer expires retransmit as specified by BACnet.
error	Test complete, no action needed
reject	Retransmit as specified by BACnet
simpleACK	Continue to next step

The lower tester now issues a ReadProperty service request with the following parameter.

'Read Access Specification' = ("Zone1ALARM", Acknowledged)

IUT:

Process the AcknowledgeAlarm service indication and the ReadProperty service indication as specified in BACnet.

Upper Tester:

No action is required for this test.

Pass Result:

A correctly formed Result(+) response primitive shall be returned by the IUT as a result of the AcknowledgeAlarm service request. A correctly formed Result(+) response primitive shall be returned by the IUT conveying the following argument:

'Read Access Result' = ("Zone1ALARM", Acknowledged, TRUE)

Fail Result:

If a correct Result(+) confirm primitive is not received as a result of the AcknowledgeAlarm service request the test fails. If the AcknowledgeAlarm service request is confirmed but the response to the subsequent ReadProperty request indicates that the value of the Acknowledged property has not been correctly changed the test fails.

Inconclusive Result:

If the AcknowledgeAlarm service request is confirmed but a value is not returned from the ReadProperty service request because of an error, a reject, or no response is received, then the test is inconclusive.

7.1.1.2 File Access Services

At the present time the File Access services are in a considerable state of flux with some radically different proposals being considered by the SPC. Defining test cases for these services will be delayed until the situation becomes more clear.

7.1.1.3 Object Access Services

Test B.1.3.1.1 Support for the AddListElement Service

Test Purpose:

Verify that the IUT can correctly respond to a valid AddListElement service request.

BACnet Clause Reference: 6.5.1

Test Description:

The lower tester issues an AddListElement service request addressed to the IUT. When the lower tester receives a confirmation (after retries if needed) a ReadProperty service request is issued. The response to the ReadProperty service request is analyzed to determine if the AddListElement service request was successfully completed.

Expected Result:

The IUT issues a Result(+) response acknowledging the completion of the AddListElement request. The IUT issues a Result(+) response to the ReadProperty service request which conveys the new value of the property affected by the AddListElement indicating that the new elements have been added to the list.

Lower Tester:

The lower tester issues an AddListElement confirmed-request PDU which conveys the parameters shown below.

```
'Object Identifier'    = "TestGroup"
'Property Identifier' =  List_of_Access_Specifications
'List of Elements'     = (("HighPressSwitch", (Present_Value, Polarity, Reliability)),
                                ("Floor3ExhaustFan", (Present_Value, Polarity, Reliability))
```

After the request is issued several things can happen and the lower tester must respond as indicated below.

IUT Response PDU	Lower Tester Action
none	When a timer expires retransmit as specified by BACnet
error	Test complete, no action needed
reject	Retransmit as specified by BACnet
simpleACK	Continue to the next step

At this point the lower tester issues a ReadProperty confirmed-request PDU which conveys the following parameter.

'Read Access Specification' = ("TestGroup", List_of_Access_Specifications)

After the request is issued several things can happen and the lower tester must respond as indicated below.

IUT Response PDU	Lower Tester Action
none	When a timer expires retransmit as specified by BACnet
error	Test complete, no action needed
reject	Retransmit as specified by BACnet
simpleACK	Restore the property to its original value

The List_of_Access_Specifications is restored to its original value by issuing a WriteProperty confirmed-request PDU conveying the following parameter.

'Write Access Specification' = (NULL, "TestGroup", ((("1AH1MAT",(Min_Pres, Max_Pres, Description)), ("MA_SetPoint",(Present_Value, Units)), ("Alarm_Consoles",(Description, Device_List)))

IUT:

Process the AddListElement request and the subsequent ReadProperty and WriteProperty requests and issue responses as specified in BACnet.

Upper Tester:

No action is required for this test.

Pass Result:

A correctly formed Result(+) response primitive shall be returned by the IUT in response to the AddListElement service request. A correctly formed complexACK-PDU shall be returned by the IUT in response to the ReadProperty request which conveys the following parameter.

'Read Access Result' = (NULL, "TestGroup", ((("1AH1MAT",(Min_Pres, Max_Pres, Description)), ("MA_SetPoint",(Present_Value, Units)), ("Alarm_Consoles",(Description, Device_List)), ("HighPressSwitch", (Present_Value, Polarity, Reliability)), ("Floor3ExhaustFan", (Present_Value, Polarity, Reliability))

A correctly formed Result(+) response primitive shall be returned by the IUT in response to the WriteProperty request.

Fail Result:

If the Result(+) response is returned but is not correctly formed the test fails. If the AddListElement request is confirmed but the response to the subsequent ReadProperty request indicates that the value of the property has not been correctly changed the test fails.

Inconclusive Result:

If the AddListElement service request is confirmed but a value is not returned from the ReadProperty service because of an error, a reject, or no response is received, then the test result is inconclusive. Further tests may be needed to determine if the AddListElement of the ReadPropery service has failed. The results are also inconclusive if a reply is not received for one or more of the lower tester's requests after all retransmissions have been exhausted.

Test B.1.3.2.1 Support for the CreateEnrollmentObject Service With No Optional Parameters

Test Purpose:[1]

Verify that the IUT can correctly respond to a valid CreateEnrollmentObject request with no optional parameters.

BACnet Clause Reference: 6.5.2

Test Description:

The lower tester issues a CreateEnrollmentObject service request addressed to the IUT. When the lower tester receives a confirmation (after retries if needed) a ReadMultipleProperty service request or a series of ReadProperty service requests is issued. The response to the ReadMultipleProperty of the series of ReadProperty requests is analyzed to determine if the new Event Enrollment Object was in fact created.

Expected Result:

The IUT issues a Result(+) response acknowledging the completion of the CreateEnrollmentObject service request. The IUT issues a Result(+) response to the ReadMultipleProperty request or a series of Result(+) responses to the ReadProperty requests conveying the property values of the newly created Event Enrollment object.

Lower Tester:

The lower tester issues a CreateEnrollmentObject confirmed-request PDU which conveys the following parameters.

Object Identifier	= "TestAlarm"
Event Type	= ALARM
Property Reference List	= ("MA_SetPoint", Present_Value)
Notification Rules	= (NORMAL_TO_ALARM, 2, 3, TRUE, "This is a test alarm")
Parameter List	= (50, 60, 1)

After the request is issued several things can happen and the lower tester must respond as indicated below.

IUT Response PDU	Lower Tester Action
none	When a timer expires retransmit as specified by BACnet
error	Test complete, no action needed
reject	Retransmit as specified by BACnet
simpleACK	Continue to the next step

At this point the lower tester attempts to verify that the Event Enrollment was successfully created by reading its properties.

Option 1 - Issue a ReadMultipleProperty service request with the following argument:

'List of Read Access Results' = (TestAlarm", ALL)

Option 2 - Issue a sequence of ReadProperty requests, one for each of the eleven properties of the object. The sequence of arguments is shown below.

'Read Access Specification' = ("TestAlarm", Object_Identifier)

---

[1]This test is based on DRAFT 2 of the CreateEnrollmentObject Service in [7].

'Read Access Specification' = ("TestAlarm", Object_Type)
'Read Access Specification' = ("TestAlarm", Event_Type)
'Read Access Specification' = ("TestAlarm", Property_Reference)
'Read Access Specification' = ("TestAlarm", State)
'Read Access Specification' = ("TestAlarm", Enabled)
'Read Access Specification' = ("TestAlarm", Acknowledged)
'Read Access Specification' = ("TestAlarm", Notification_Rules)
'Read Access Specification' = ("TestAlarm", Parameter_List)
'Read Access Specification' = ("TestAlarm", Confirmed_Recipient_List)
'Read Access Specification' = ("TestAlarm", Unconfirmed_Recipient_List)

IUT:

Process the CreateEnrollmentObject request and the ReadMultipleProperty or the sequence of ReadProperty requests as specified in BACnet.

Upper Tester:

No action is required for this test.

Pass Result:

The IUT responds to the CreateEnrollmentObject service with a Result(+). If the lower tester follows this with a ReadMultipleProperty request, the IUT responds with a Result(+) containing the following argument:

'List of Read Access Results' = ("TestAlarm", EVENT_ENROLLMENT, ALARM, ("MA_SetPoint", Present_Value), ALARM, TRUE, FALSE, (NORMAL_TO_ALARM, 2, 3, TRUE, "This is a test alarm"), (50, 60, 1), NULL, NULL)

If the lower tester follows instead with the series of ReadProperty requests then a sequence of Result(+) confirm primitives are received by the lower tester, one for each request issued. Each Result(+) primitive will contain one argument. The sequence will be:

'Read Access Result' = "TestAlarm"
'Read Access Result' = EVENT_ENROLLMENT
'Read Access Result' = ALARM
'Read Access Result' = ("MA_SetPoint", Present_Value)
'Read Access Result' = ALARM
'Read Access Result' = TRUE
'Read Access Result' = FALSE
'Read Access Result' = (NORMAL_TO_ALARM), 2, 3, TRUE, "This is a test alarm")
'Read Access Result' = (50, 60, 1)
'Read Access Result' = NULL
'Read Access Result' = NULL

Fail Result:

If the correct response to the CreateEnrollmentObject service request is not received the test fails. If the CreateEnrollmentObject request is properly confirmed but the response to the subsequent ReadMultipleProperty or series of ReadProperty requests indicates that the Event Enrollment object was not properly created the test fails.

Inconclusive Result:

If the CreateEnrollmentObject service request is confirmed but values are not returned from the ReadMultipleProperty or ReadProperty service requests because of an error, a reject, or no response is

received, then the test results are inconclusive. Further tests may be needed to determine if the CreateEnrollmentObject, ReadMultipleProperty, or the ReadProperty service has failed.

Test B.1.3.2.2 Support for the CreateEnrollmentObject Service With All Optional Parameters

Test Purpose:[1]

Verify that the IUT can correctly respond to a valid CreateEnrollmentObject request with all optional parameters.

BACnet Clause Reference: 6.5.2

Test Description:

The lower tester issues a CreateEnrollmentObject service request addressed to the IUT. When the lower tester receives a confirmation (after retries if needed) a ReadMultipleProperty service request or a series of ReadProperty service requests is issued. The response to the ReadMultipleProperty of the series of ReadProperty requests is analyzed to determine if the new Event Enrollment Object was in fact created.

Expected Result:

The IUT issues a Result(+) response acknowledging the completion of the CreateEnrollmentObject service request. The IUT issues a Result(+) response to the ReadMultipleProperty request or a series of Result(+) responses to the ReadProperty requests conveying the property values of the newly created Event Enrollment object.

Lower Tester:

The lower tester issues a CreateEnrollmentObject confirmed-request PDU which conveys the following parameters.

Object Identifier	= "TestAlarm"
Event Type	= ALARM
Property Reference List	= ("MA_SetPoint", Present_Value)
Enabled	= TRUE
Notification Rules	= (NORMAL_TO_ALARM, 2, 3, TRUE, "This is a test alarm")
Parameter List	= (50, 60, 1)
Confirmed Recipient List	= lower tester
Unconfirmed Recipient List	= GLOBAL

After the request is issued several things can happen and the lower tester must respond as indicated below.

IUT Response PDU	Lower Tester Action
none	When a timer expires retransmit as specified by BACnet
error	Test complete, no action needed
reject	Retransmit as specified by BACnet
simpleACK	Continue to the next step

At this point the lower tester attempts to verify that the Event Enrollment was successfully created by

---

[1]This test is based on DRAFT 2 of the CreateEnrollmentObject Service in [7].

reading its properties.

Option 1 - Issue a ReadMultipleProperty service request with the following argument:

'List of Read Access Results' = (TestAlarm", ALL)

Option 2 - Issue a sequence of ReadProperty requests, one for each of the eleven properties of the object. The sequence of arguments is shown below.

'Read Access Specification' = ("TestAlarm", Object_Identifier)
'Read Access Specification' = ("TestAlarm", Object_Type)
'Read Access Specification' = ("TestAlarm", Event_Type)
'Read Access Specification' = ("TestAlarm", Property_Reference)
'Read Access Specification' = ("TestAlarm", State)
'Read Access Specification' = ("TestAlarm", Enabled)
'Read Access Specification' = ("TestAlarm", Acknowledged)
'Read Access Specification' = ("TestAlarm", Notification_Rules)
'Read Access Specification' = ("TestAlarm", Parameter_List)
'Read Access Specification' = ("TestAlarm", Confirmed_Recipient_List)
'Read Access Specification' = ("TestAlarm", Unconfirmed_Recipient_List)

IUT:

Process the CreateEnrollmentObject request and the ReadMultipleProperty or the sequence of ReadProperty requests as specified in BACnet.

Upper Tester:

No action is required for this test.

Pass Result:

The IUT responds to the CreateEnrollmentObject service with a Result(+). If the lower tester follows this with a ReadMultipleProperty request the IUT responds with a Result(+) containing the following argument:

'List of Read Access Results' = ("TestAlarm", EVENT_ENROLLMENT, ALARM, ("MA_SetPoint", Present_Value), ALARM, TRUE, FALSE, (NORMAL_TO_ALARM, 2, 3, TRUE, "This is a test alarm"), (50, 60, 1), NULL, NULL)

If the lower tester follows instead with the series of ReadProperty requests then a sequence of Result(+) confirm primitives are received by the lower tester, one for each request issued. Each Result(+) primitive will contain one argument. The sequence will be:

'Read Access Result' = "TestAlarm"
'Read Access Result' = EVENT_ENROLLMENT
'Read Access Result' = ALARM
'Read Access Result' = ("MA_SetPoint", Present_Value)
'Read Access Result' = ALARM
'Read Access Result' = TRUE
'Read Access Result' = FALSE
'Read Access Result' = (NORMAL_TO_ALARM), 2, 3, TRUE, "This is a test alarm")
'Read Access Result' = (50, 60, 1)
'Read Access Result' = lower tester
'Read Access Result' = GLOBAL

Fail Result:

If the correct response to the CreateEnrollmentObject service request is not received the test fails. If the CreateEnrollmentObject request is properly confirmed but the response to the subsequent ReadMultipleProperty or series of ReadProperty requests indicates that the Event Enrollment object was not properly created the test fails.

Inconclusive Result:

If the CreateEnrollmentObject service request is confirmed but values are not returned from the ReadMultipleProperty or ReadProperty service requests because of an error, a reject, or no response is received, then the test results are inconclusive. Further tests may be needed to determine if the CreateEnrollmentObject, ReadMultipleProperty, or the ReadProperty service has failed.

Test B.1.3.3.1 Support for the CreateGroupObject Service

Test Purpose:

Verify that the IUT can correctly respond to a valid CreateGroupObject request.

BACnet Clause Reference: 6.5.3

Test Description:

The lower tester issues a CreateGroupObject service request addressed to the IUT. When the lower tester receives a confirmation (after retries if needed) a ReadMultipleProperty service request or a series of ReadProperty service requests is issued. The response to the ReadMultipleProperty of the series of ReadProperty requests is analyzed to determine if the new Group Object was in fact created.

Expected Result:

The IUT issues a Result(+) response acknowledging the completion of the CreateGroupObject service request. The IUT issues a Result(+) response to the ReadMultipleProperty request or a series of Result(+) responses to the ReadProperty requests conveying the property values of the newly created Event Enrollment object.

Lower Tester:

The lower tester issues a CreateGroupObject confirmed-request PDU which conveys the following parameters.

Object Identifier	= "Test_Group"
List_of_Access_Specifications	= (("1AH1MAT",(Min_Pres, Max_Pres, Description)), ("MA_SetPoint",(Present_Value, Units)), ("Alarm_Consoles",(Description, Device_List)))

After the request is issued several things can happen and the lower tester must respond as indicated below.

IUT Response PDU	Lower Tester Action
none	When a timer expires retransmit as specified by BACnet
error	Test complete, no action needed
reject	Retransmit as specified by BACnet
simpleACK	Continue to the next step

At this point the lower tester attempts to verify that the Group object was successfully created by reading its properties.

Option 1 - Issue a ReadMultipleProperty service request with the following argument:

'List of Read Access Results' = (Test_Group", ALL)

Option 2 - Issue a sequence of ReadProperty requests, one for each of the eleven properties of the object. The sequence of arguments is shown below.

'Read Access Specification' = ("Test_Group", Object_Identifier)
'Read Access Specification' = ("Test_Group", Object_Type)
'Read Access Specification' = ("Test_Group", List_Of_Access_Specifications)
'Read Access Specification' = ("Test_Group", Present_Value)

IUT:

Process the CreateGroupObject request and the ReadMultipleProperty or the sequence of ReadProperty requests as specified in BACnet.

Upper Tester:

No action is required for this test.

Pass Result:

The IUT responds to the CreateGroupObject service with a Result(+). If the lower tester follows this with a ReadMultipleProperty request the IUT responds with a Result(+) containing the following argument:

'List of Read Access Results' = ("Test_Group", GROUP, (("1AH1MAT", (Min_Pres, Max_Pres, Description)), ("MA_SetPoint", (Present_Value, Units)), ("Alarm Consoles", (Description, Device_List)), (-50, 250, "Mixed Air Temperature", 58.0, "DEGF", "Building ABC Host Computer Systems", "Node 5", "Node 7"))

If the lower tester follows instead with the series of ReadProperty requests then a sequence of Result(+) confirm primitives are received by the lower tester, one for each request issued. Each Result(+) primitive will contain one argument. The sequence will be:

'Read Access Result' = "Test_Group"
'Read Access Result' = GROUP
'Read Access Result' = (("1AH1MAT", (Min_Pres, Max_Pres, Description)),
("MA_SetPoint", (Present_Value, Units)),
("Alarm Consoles",(Description,Device_List)),
'Read Access Result' = (-50, 250, "Mixed Air Temperature", "Alarm Consoles"
58.0, "DEGF", "Building ABC Host Computer Systems","Node 5","Node 7"))

Fail Result:

If the correct response to the CreateGroupObject service request is not received the test fails. If the CreateGroupObject request is properly confirmed but the response to the subsequent ReadMultipleProperty or series of ReadProperty requests indicates that the Group object was not properly created the test fails.

Inconclusive Result:

If the CreateGroupObject service request is confirmed but values are not returned from the ReadMultipleProperty or ReadProperty service requests because of an error, a reject, or no response is received, then the test results are inconclusive. Further tests may be needed to determine if the CreateGroupObject, ReadMultipleProperty, or the ReadProperty service has failed.

Test B.1.3.4.1 Support for the DeleteObject Service

Test Purpose:

Verify that the IUT can correctly respond to a valid DeleteObject service request.

BACnet Clause Reference: 6.5.4

Test Description:

The lower tester issues a DeleteObject service request addressed to the IUT. The ReadProperty service is then used to verify that the object has in fact been deleted.

Expected Result:

The IUT issues a Result(+) response to the DeleteObject service request. The IUT responds to the subsequent ReadProperty with a Result(-) response conveying an error class of OBJECT and an error code of UNKNOWN _OBJECT.

Lower Tester:

The lower tester issues a DeleteObject confirmed-request-PDU which conveys the parameter 'Object Identifier' = "Test_Group". After the request is issued several things can happen and the lower tester must respond as indicated below.

IUT Response PDU	Lower Tester Action
none	When a timer expires retransmit as specified by BACnet
error	Test complete, no action needed
reject	Retransmit as specified by BACnet
simpleACK	Continue to next step

At this point the lower tester issues a ReadProperty confirmed-request PDU which conveys the following parameter.

'Read Access Specification' = ("Test_Group")

After the request is issued several things can happen and the lower tester must respond as indicated below.

IUT Response PDU	Lower Tester Action
none	When a timer expires retransmit as specified by BACnet
error	Test complete, no action needed
reject	Retransmit as specified by BACnet
complexACK	Test complete, no action needed

IUT:

Process the DeleteObject service request and the ReadProperty service request as specified by BACnet.

Upper Tester:

No action needed for this test.

Pass Result:

The lower tester should receive a Result(+) confirm from the IUT confirming the DeleteObject service and a Result(-) confirm confirming the ReadProperty service request conveying and error class of OBJECT and error code of UNKNOWN_OBJECT.

Fail Result:

If a response to the DeleteObject service request is received but it is not a correct Result(+) the test fails. If a correct response to the DeleteObject service is received but a Result(+) confirm or a Result(-) confirm which is not as specified above is received in response to the ReadProperty service then the test fails. The test also fails if the DeleteObject service request is repeatedly rejected.

Inconclusive Results:

The results are inconclusive if the IUT does not respond. The communication link should be checked to ensure that the communication has not been lost due to a lower layer failure. If this is the case the test must be retried. If this is not the case the test has failed.

Test B.1.3.6.1 Support for the ReadMultipleProperty With No 'Return Read Access with Result' Parameter

Test Purpose:

Verify that the IUT can correctly respond to a valid ReadMultipleProperty service request with no 'Return Read Access Specification with Result' parameter. All of the allowed combinations for the 'List of Access Specifications' are included in this test.

BACnet Clause Reference: 6.5.6

Test Description:

The lower tester issues a ReadProperty service request addressed to the IUT with no 'Return Read Access Specification' parameter and the response is analyzed based on the standard object configuration.

note: This test assumes that the IUT supports Analog Input, Analog Output, and Binary Input object-types. If this assumption is not valid then the details of the 'List of Access Specifications' must be modified.

Expected Result:

The IUT issues a Result(+) response PDU which correctly encodes the valued specified in the ReadMultipleProperty request.

Lower Tester:

The lower tester issues a ReadMultipleProperty confirmed-request PDU with the parameters shown below.

'List of Read Access Specifications' = ((lower tester, "1AH1MAT", Present_Value, Reliability, Description), (NULL, "1AH1DMPR", Present_Value, Status_Flags), (NULL, "HighPressSwitch", NULL))

After the request is issued several things can happen and the lower tester must respond as indicated below.

IUT Response PDU	Lower Tester Action
none	When a timer expires retransmit as specified by BACnet
error	Test complete, no action needed
reject	Retransmit as specified by BACnet
complexACK	Test complete, no action needed

IUT:

Process the confirmed-service request as specified by BACnet.

Upper Tester:

No action required for this case.

Pass Result:

A correctly formed complexACK-PDU shall be returned by the IUT to the lower tester conveying the following Result(+) parameter.

'Read Access Result' = (58.1, RELIABLE, "Mixed Air Temperature", 75.0, {FALSE, TRUE, FALSE, FALSE}, ACTIVE)

This result may come after retransmitting one or more times, up to the limit specified in BACnet, and the test results are still considered a pass. The fact that retransmissions were needed should however be noted because this may be useful diagnostic information for uncovering a conformance problem with the IUT.

Fail Result:

If a complexACK-PDU is returned that must be rejected by the lower tester because it is incorrectly formed, or if it is correctly formed but does not covey the information described above the test result is a failure. The test result is also a failure if the IUT returns an error-PDU or rejects the original request and all retries. It may be possible to determine from the test results that failure was due to the fact that the IUT does not correctly implement the optional 'BACnet Device Name' or the optional 'List of Property Identifiers'.

Inconclusive Result:

The results inconclusive if the IUT does not respond. The communication link should be checked to ensure that communication has not been lost due to a lower layer failure. If this is the case the test must be retried. If this is not the case then this test has failed.

Test B.1.3.6.2 Support for the ReadMultipleProperty With 'Return Read Access with Result' = FALSE

Test Purpose:

Verify that the IUT can correctly respond to a valid ReadMultipleProperty service request with 'Return Read Access Specification with Result' = FALSE. All of the allowed combinations for the 'List of Access Specifications' are included in this test.

BACnet Clause Reference: 6.5.6

Test Description:

The lower tester issues a ReadProperty service request addressed to the IUT with 'Return Read Access Specification with Result'= FALSE and the response is analyzed based on the standard object configuration.

note: This test assumes that the IUT supports Analog Input, Analog Output, and Binary Input object-types. If this assumption is not valid then the details of the 'List of Access Specifications' must be modified.

Expected Result:

The IUT issues a Result(+) response PDU which correctly encodes the valued specified in the ReadMultipleProperty request.

Lower Tester:

The lower tester issues a ReadMultipleProperty confirmed-request PDU with the parameters shown below.

'List of Read Access Specifications' = ((lower tester, "1AH1MAT", Present_Value, Reliability,

Description),(NULL,"1AH1DMPR",Present_Value,Status_Flags),(NULL,"HighPressSwitch",
NULL))
'Return Read Access Specifications with Result' = FALSE

After the request is issued several things can happen and the lower tester must respond as indicated below.

IUT Response PDU	Lower Tester Action
none	When a timer expires retransmit as specified by BACnet
error	Test complete, no action needed
reject	Retransmit as specified by BACnet
complexACK	Test complete, no action needed

IUT:

Process the confirmed-service request as specified by BACnet.

Upper Tester:

No action required for this case.

Pass Result:

A correctly formed complexACK-PDU shall be returned by the IUT to the lower tester conveying the following Result(+) parameter.

'Read Access Result' = (58.1, RELIABLE, "Mixed Air Temperature", 75.0, {FALSE, TRUE, FALSE, FALSE}, ACTIVE)

This result may come after retransmitting one or more times, up to the limit specified in BACnet, and the test results are still considered a pass. The fact that retransmissions were needed should however be noted because this may be useful diagnostic information for uncovering a conformance problem with the IUT.

Fail Result:

If a complexACK-PDU is returned that must be rejected by the lower tester because it is incorrectly formed, or if it is correctly formed but does not covey the information described above, the test result is a failure. The test result is also a failure if the IUT returns an error-PDU or rejects the original request and all retries. It may be possible to determine from the test results that failure was due to the fact that the IUT does not correctly implement the optional 'BACnet Device Name' or the optional 'List of Property Identifiers'. If the 'Read Access Specifications' are returned with the result, the IUT does not properly implement the 'Return Read Access Specifications with Result' parameter.

Inconclusive Result:

The results inconclusive if the IUT does not respond. The communication link should be checked to ensure that communication has not been lost due to a lower layer failure. If this is the case the test must be retried. If this is not the case then this test has failed.

Test B.1.3.6.3 Support for the ReadMultipleProperty With 'Return Read Access with Result' = TRUE

Test Purpose:

Verify that the IUT can correctly respond to a valid ReadMultipleProperty service request with 'Return Read Access Specification with Result' = TRUE. All of the allowed combinations for the 'List of Access Specifications' are included in this test.

BACnet Clause Reference: 6.5.6

Test Description:

The lower tester issues a ReadProperty service request addressed to the IUT with 'Return Read Access Specification with Result' = TRUE and the response is analyzed based on the standard object configuration.

note: This test assumes that the IUT supports Analog Input, Analog Output, and Binary Input object-types. If this assumption is not valid then the details of the 'List of Access Specifications' must be modified.

Expected Result:

The IUT issues a Result(+) response PDU which correctly encodes the valued specified in the ReadMultipleProperty request.

Lower Tester:

The lower tester issues a ReadMultipleProperty confirmed-request PDU with the parameters shown below.

'List of Read Access Specifications' = ((lower tester, "1AH1MAT", Present_Value, Reliability, Description), (NULL, "1AH1DMPR", Present_Value, Status_Flags), (NULL, "HighPressSwitch", NULL))
'Return Read Access Specifications with Result' = TRUE

After the request is issued several things can happen and the lower tester must respond as indicated below.

IUT Response PDU	Lower Tester Action
none	When a timer expires retransmit as specified by BACnet
error	Test complete, no action needed
reject	Retransmit as specified by BACnet
complexACK	Test complete, no action needed

IUT:

Process the confirmed-service request as specified by BACnet.

Upper Tester:

No action required for this case.

Pass Result:

A correctly formed complexACK-PDU shall be returned by the IUT to the lower tester conveying the following Result(+) parameter.

'Read Access Result' = (lower tester, "1AH1MAT", Present_Value, 58.1, Reliability, RELIABLE, Description, "Mixed Air Temperature", "1AH1DMPR", Present_Value, 75.0, Status_Flags, {FALSE, TRUE, FALSE, FALSE}, "HighPressSwitch", Present_Value, ACTIVE)

This result may come after retransmitting one or more times, up to the limit specified in BACnet, and the test results are still considered a pass. The fact that retransmissions were needed should however be noted because this may be useful diagnostic information for uncovering a conformance problem with the IUT.

Fail Result:

If a complexACK-PDU is returned that must be rejected by the lower tester because it is incorrectly formed, or if it is correctly formed but does not covey the information described above the test result is a failure. The test result is also a failure if the IUT returns an error-PDU or rejects the original request and all retries. It may be possible to determine from the test results that failure was due to the fact that the IUT does not correctly implement the optional 'BACnet Device Name' or the optional 'List of Property

Identifiers'. If the 'Read Access Specifications' are not returned with the result, the IUT does not properly implement the 'Return Read Access Specifications with Result' parameter.

Inconclusive Result:

The results inconclusive if the IUT does not respond. The communication link should be checked to ensure that communication has not been lost due to a lower layer failure. If this is the case the test must be retried. If this is not the case then this test has failed.

Test B.1.3.7.1 Support for the ReadProperty Service Using All Default Parameters

Test Purpose:

Verify that the IUT can correctly respond to a valid ReadProperty service request with default values for the 'Object Identifier' and 'Property Identifier' parameters in the 'Read Access Specification'.

BACnet Clause Reference: 6.5.7

Test Description:

The lower tester issues a ReadProperty service request addressed to the IUT using all default values for the 'Read Access Specification' and the response is analyzed based on the standard object configuration.

Expected Result:

The IUT issues a Result(+) response PDU which correctly encodes the value for the System_Status property of the Device object (the default property of the default object).

Lower Tester:

The lower tester issues a ReadProperty confirmed-request PDU with default values for the Read Access Specification. After the request is issued several things can happen and the lower tester must respond as indicated below.

IUT Response PDU	Lower Tester Action
none	When a timer expires retransmit as specified by BACnet
error	Test complete, no action needed
reject	Retransmit as specified by BACnet
complexACK	Test complete, no action needed

IUT:

Process the confirmed-service request as specified by BACnet.

Upper Tester:

No action required for this case.

Pass Result:

A correctly formed complexACK-PDU shall be returned by the IUT to the lower tester conveying the following Result(+) parameter.

'Read Access Result' = (device object ID, System_Status, OPERATIONAL)

This result may come after retransmitting one or more times, up to the limit specified in BACnet, and the test results are still considered a pass. The fact that retransmissions were needed should however be noted because this may be useful diagnostic information for uncovering a conformance problem with the IUT.

Fail Result:

If a complexACK-PDU is returned that must be rejected by the lower tester because it is incorrectly formed, or if it is correctly formed but does not covey the information described above the test result is a failure. The test result is also a failure if the IUT returns an error-PDU or rejects the original request and all retries.

Inconclusive Result:

The result is inconclusive if the IUT does not respond. The communication link should be checked to ensure that communication has not been lost due to a lower layer failure. If this is the case the test must be retried. If this is not the case then this test has failed.

Test B.1.3.7.2 Support for the ReadProperty Service Using Default Object Identifier

Test Purpose:

Verify that the IUT can correctly respond to a valid ReadProperty service request with a default value for the 'Object Identifier' in the 'Read Access Specification'.

BACnet Clause Reference: 6.5.7

Test Description:

The lower tester issues a ReadProperty service request addressed to the IUT specifying a property identifier and the default object identifier for the 'Read Access Specification'. The response is analyzed based on the standard object configuration.

Expected Result:

The IUT issues a ComplexACK-PDU conveying a Result(+) response which correctly encodes the value for the specified property of the Device object (the default object).

Lower Tester:

The lower tester issues a ReadProperty confirmed-request PDU specifying the Protocol_Version (or another DEVICE object property) property identifier and no device identifier for the 'Read Access Specification'. After the request is issued several things can happen and the lower tester must respond as indicated below.

IUT Response PDU	Lower Tester Action
none	When a timer expires retransmit as specified by BACnet
error	Test complete, no action needed
reject	Retransmit as specified by BACnet
complexACK	Test complete, no action needed

IUT:

Process the confirmed-service request as specified by BACnet.

Upper Tester:

No action required for this case.

Pass Result:

A correctly formed complexACK-PDU shall be returned by the IUT to the lower tester conveying the following Result(+) parameter.

'Read Access Result' = (device object ID, Protocol_Version, 1.0)

This result may come after retransmitting one or more times, up to the limit specified in BACnet, and the test results are still considered a pass. The fact that retransmissions were needed should however be noted because this may be useful diagnostic information for uncovering a conformance problem with the IUT.

Fail Result:

If a complexACK-PDU is returned that must be rejected by the lower tester because it is incorrectly formed, or if it is correctly formed but does not covey the information described above the test result is a failure. The test result is also a failure if the IUT returns an error-PDU or rejects the original request and all retries.

Inconclusive Result:

The result is inconclusive if the IUT does not respond. The communication link should be checked to ensure that communication has not been lost due to a lower layer failure. If this is the case the test must be retried. If this is not the case then this test has failed.

Test B.1.3.7.3 Support for the ReadProperty Service Using Default Property Identifier

Test Purpose:

Verify that the IUT can correctly respond to a valid ReadProperty service request with a default value for the 'Property Identifier' in the 'Read Access Specification'.

BACnet Clause Reference 6.5.7

Test Description:

The lower tester issues a ReadProperty service request addressed to the IUT specifying an object identifier and the default property identifier for the 'Read Access Specification'. The response is analyzed based on the standard object configuration.

Expected Result:

The IUT issues a BACnet-ComplexACK-PDU conveying a Result(+) response which correctly encodes the value for the default property of the specified object.

Lower Tester:

The lower tester issues a ReadProperty confirmed-request PDU specifying the DEVICE object identifier (or some other object identifier) but no property identifier in the 'Read Access Specification'. After the request is issued several things can happen and the lower tester must respond as indicated below.

IUT Response PDU	Lower Tester Action
none	When a timer expires retransmit as specified by BACnet
error	Test complete, no action needed
reject	Retransmit as specified by BACnet
complexACK	Test complete, no action needed

IUT:

Process the confirmed-service request as specified by BACnet.

Upper Tester:

No action required for this case.

Pass Result:

A correctly formed complexACK-PDU shall be returned by the IUT to the lower tester conveying the following Result(+) parameter.

'Read Access Result' = (device object ID, System_Status, OPERATIONAL)

This result may come after retransmitting one or more times, up to the limit specified in BACnet, and the test results are still considered a pass. The fact that retransmissions were needed should however be noted because this may be useful diagnostic information for uncovering a conformance problem with the IUT.

Fail Result:

If a complexACK-PDU is returned that must be rejected by the lower tester because it is incorrectly formed, or if it is correctly formed but does not covey the information described above the test result is a failure. The test result is also a failure if the IUT returns an error-PDU or rejects the original request and all retries.

Inconclusive Result:

The result is inconclusive if the IUT does not respond. The communication link should be checked to ensure that communication has not been lost due to a lower layer failure. If this is the case the test must be retried. If this is not the case then this test has failed.

Test B.1.3.7.4 Support for the ReadProperty Service With No Default Parameters

Test Purpose:

Verify that the IUT can correctly respond to a valid ReadProperty service request with no default values.

BACnet Clause Reference: 6.5.7

Test Description:

The lower tester issues a ReadProperty service request addressed to the IUT specifying an object identifier and a property identifier for the 'Read Access Specification'. The response is analyzed based on the standard object configuration.

Expected Result:

The IUT issues a BACnet-ComplexACK-PDU conveying a Result(+) response which correctly encodes the value for the specified object and property.

Lower Tester:

The lower tester issues a ReadProperty confirmed-request PDU specifying "1AH1MAT" as the object identifier (or some other object identifier) and State (or some other property identifier for this object) in the 'Read Access Specification'. After the request is issued several things can happen and the lower tester must respond as indicated below.

IUT Response PDU	Lower Tester Action
none	When a timer expires retransmit as specified by BACnet
error	Test complete, no action needed
reject	Retransmit as specified by BACnet
complexACK	Test complete, no action needed

IUT:

Process the confirmed-service request as specified by BACnet.

Upper Tester:

No action required for this case.

Pass Result:

A correctly formed complexACK-PDU shall be returned by the IUT to the lower tester conveying the following Result(+) parameter.

'Read Access Result' = ("1AH1MAT", State, NORMAL)

This result may come after retransmitting one or more times, up to the limit specified in BACnet, and the test results are still considered a pass. The fact that retransmissions were needed should however be noted because this may be useful diagnostic information for uncovering a conformance problem with the IUT.

Fail Result:

If a complexACK-PDU is returned that must be rejected by the lower tester because it is incorrectly formed, or if it is correctly formed but does not covey the information described above the test result is a failure. The test result is also a failure if the IUT returns an error-PDU or rejects the original request and all retries.

Inconclusive Result:

The result is inconclusive if the IUT does not respond. The communication link should be checked to ensure that communication has not been lost due to a lower layer failure. If this is the case the test must be retried. If this is not the case then this test has failed.

Test B.1.3.10.1 Support for the WriteMultipleProperty Service Using Default Property Identifier

Test Purpose:

Verify that the IUT can correctly respond to a valid WriteMultipleProperty service request with a default value for the 'Property Identifier' parameter in the 'Write Access Specification'.

BACnet Clause Reference: 6.5.10

Test Description:

The lower tester issues a WriteMultipleProperty service request addressed to the IUT using a default value for the 'Property Identifier' parameter. When the WriteMultipleProperty service request is acknowledged (possibly after retries) the lower tester issues a ReadMultipleProperty service request addressed to the IUT specifying the same objects and properties that was written to in the WriteMultipleProperty service request. The values returned from the ReadMultipleProperty request are then compared with the values that were written.

Expected Result:

After processing the WriteMultipleProperty request the IUT issues a Result(+) response primitive confirming that the write has been successfully completed. After processing the ReadMultipleProperty request the IUT issues a Result(+) response primitive which correctly conveys the new values for the specified objects and properties.

Lower Tester:

The lower tester issues a WriteMultipleProperty confirmed-request PDU with the following parameter.

List of Write Access Specification = (("Floor3ExhaustFan", NULL, ACTIVE), ("ExhaustFanEnable", NULL, INACTIVE))

After the request is issued several things can happen and the lower tester must respond as indicated below.

IUT Response PDU	Lower Tester Action
none	When a timer expires retransmit as specified by BACnet
error	Test complete, no action needed
reject	Retransmit as specified by BACnet
simpleACK	Continue to the next step

At this point the lower tester issues a ReadMultiplePropery confirmed-request PDU with the following parameter.

'List of Read Access Specifications' = ((NULL, "Floor3ExhaustFan", NULL), (NULL, "ExhaustFanEnable", NULL))

After the request is issued several things can happen and the lower tester must respond as indicated below.

IUT Response PDU	Lower Tester Action
none	When a timer expires retransmit as specified by BACnet
error	Test complete, no action needed
reject	Retransmit as specified by BACnet
ComplexACK	Evaluate response

The lower tester then restores the Present_Value of "Floor3ExhaustFan" and "ExhaustFanEnable" by writing to them again.

IUT:

Process the confirmed-service requests as specified by BACnet.

Upper Tester:

No action required for this test case.

Pass Result:

A correctly formed Result(+) response primitive shall be returned by the IUT in response to the WriteMultipleProperty request. A correctly formed complexACK-PDU shall be returned by the IUT in response to the ReadMultipleProperty request which conveys the parameter 'Read Access Result' = (ACTIVE, INACTIVE).

Either or both of these results may come after retransmitting one or more times, up to the limit specified in BACnet, and the test results are still considered a pass. The fact that retransmissions were needed should however be noted because this may be useful information for uncovering a conformance problem with the IUT.

Fail Result:

If the Result(+) response to the WriteProperty request is returned but is not correctly formed the test fails. If the WriteProperty request is confirmed but the subsequent ReadMultipleProperty indicates that the value

of the property has not been correctly changed the test fails.

Inconclusive Result:

If the WriteMultipleProperty service request is confirmed but a value is not returned from the ReadMultipleProperty service because of an error, a reject, or no response is received then the test result is inconclusive. Further tests may be needed to determine if the ReadMultipleProperty or the WriteMultipleProperty service has failed. The results are also inconclusive if a reply is not received for one or more of the lower tester's requests after all retransmissions have been exhausted.

Test B.1.3.10.2 Support for the WriteMultipleProperty Service Using Explicit Property Identifiers

Test Purpose:

Verify that the IUT can correctly respond to a valid WriteMultipleProperty service request with a explicit values for the 'Property Identifier' parameter in the 'Write Access Specification'.

BACnet Clause Reference: 6.5.10

Test Description:

The lower tester issues a WriteMultipleProperty service request addressed to the IUT using a mixture of explicit and default values for the 'Property Identifier' parameters. When the WriteMultipleProperty service request is acknowledged (possibly after retries) the lower tester issues a ReadMultipleProperty service request addressed to the IUT specifying the same objects and properties that was written to in the WriteMultipleProperty service request. The values returned from the ReadMultipleProperty request is then compared with the values that were written.

Expected Result:

After processing the WriteMultipleProperty request the IUT issues a Result(+) response primitive confirming that the write has been successfully completed. After processing the ReadMultipleProperty request the IUT issues a Result(+) response primitive which correctly conveys the new values for the specified objects and properties.

Lower Tester:

The lower tester issues a WriteMultipleProperty confirmed-request PDU with the following parameter.

List of Write Access Specification = (("TestLoop", Integral_Constant, 0.5), ("TestLoop", Maximum_OutPut, 20), ("Rm208Sched", Effective_Period, ((90/9/5), (91/8/31))),("Floor3ExhaustFan",NULL,ACTIVE),("ExhaustFanEnable",NULL,INACTIVE))

After the request is issued several things can happen and the lower tester must respond as indicated below.

IUT Response PDU	Lower Tester Action
none	When a timer expires retransmit as specified by BACnet
error	Test complete, no action needed
reject	Retransmit as specified by BACnet
simpleACK	Continue to the next step

At this point the lower tester issues a ReadMultiplePropery confirmed-request PDU with the following parameter.

'List of Read Access Specifications' = ((NULL, "TestLoop", Integral_Constant, Maximum_Output),

(NULL, "Rm208Sched", Effective_Period),(NULL, "Floor3ExhaustFan", NULL), (NULL, "ExhaustFanEnable", NULL))

After the request is issued several things can happen and the lower tester must respond as indicated below.

IUT Response PDU	Lower Tester Action
none	When a timer expires retransmit as specified by BACnet
error	Test complete, no action needed
reject	Retransmit as specified by BACnet
ComplexACK	Evaluate response

The lower tester then restores the Present_Value of "Floor3ExhaustFan" and "ExhaustFanEnable" by writing to them again.

IUT:

Process the confirmed-service requests as specified by BACnet.

Upper Tester:

No action required for this test case.

Pass Result:

A correctly formed Result(+) response primitive shall be returned by the IUT in response to the WriteMultipleProperty request. A correctly formed complexACK-PDU shall be returned by the IUT in response to the ReadMultipleProperty request which conveys the following parameter.

'Read Access Result' = (0.5, 20, ((90/9/05), (91/8/31)), ACTIVE, INACTIVE).

Either or both of these results may come after retransmitting one or more times, up to the limit specified in BACnet, and the test results are still considered a pass. The fact that retransmissions were needed should however be noted because this may be useful information for uncovering a conformance problem with the IUT.

Fail Result:

If the Result(+) response to the WriteProperty request is returned but is not correctly formed the test fails. If the WriteProperty request is confirmed but the subsequent ReadMultipleProperty indicates that the value of the property has not been correctly changed the test fails.

Inconclusive Result:

If the WriteMultipleProperty service request is confirmed but a value is not returned from the ReadMultipleProperty service because of an error, a reject, or no response is received then the test result is inconclusive. Further tests may be needed to determine if the ReadMultipleProperty or the WriteMultipleProperty service has failed. The results are also inconclusive if a reply is not received for one or more of the lower tester's requests after all retransmissions have been exhausted.

Test B.1.3.11.1 Support for the WriteProperty Service Using All Default Parameters

Test Purpose:

Verify that the IUT can correctly respond to a valid WriteProperty service request with default values for the 'BACnet Device Name' and the 'Property Identifier' parameters in the 'Write Access Specification'.

BACnet Clause Reference: 6.5.11

Test Description:

The lower tester issues a WriteProperty service request addressed to the IUT using default values for the 'Device Name' and 'Property Identifier' parameters. When the WriteProperty service request is acknowledged (possibly after retries) the lower tester issues a ReadProperty service request addressed to the IUT specifying the same object and property that was written to in the WriteProperty service request. The value returned from the ReadProperty request is then compared with the value that was written.

Expected Result:

After processing the WriteProperty request the IUT issues a Result(+) response primitive confirming that the write has been successfully completed. After processing the ReadProperty request the IUT issues a Result(+) response primitive which correctly conveys the new value for the specified object and property.

Lower Tester:

The lower tester issues a WriteProperty confirmed-request PDU with the following parameter.

Write Access Specification = (NULL, "Floor3ExhaustFan", NULL, ACTIVE)

After the request is issued several things can happen and the lower tester must respond as indicated below.

IUT Response PDU	Lower Tester Action
none	When a timer expires retransmit as specified by BACnet
error	Test complete, no action needed
reject	Retransmit as specified by BACnet
simpleACK	Continue to the next step

At this point the lower tester issues a ReadPropery confirmed-request PDU with the following parameter.

'Read Access Specification' = ("Floor3ExhaustFan", Present_Value)

After the request is issued several things can happen and the lower tester must respond as indicated below.

IUT Response PDU	Lower Tester Action
none	When a timer expires retransmit as specified by BACnet
error	Test complete, no action needed
reject	Retransmit as specified by BACnet
ComplexACK	Evaluate response

The lower tester then restores the Present_Value of "Floor3ExhaustFan" by writing to it again with a value of INACTIVE.

IUT:

Process the confirmed-service requests as specified by BACnet.

Upper Tester:

No action required for this test case.

Pass Result:

A correctly formed Result(+) response primitive shall be returned by the IUT in response to the WriteProperty request. A correctly formed complexACK-PDU shall be returned by the IUT in response to the ReadProperty request which conveys the parameter 'Read Access Result' = ("Floor3ExhaustFan", ACTIVE).

Either or both of these results may come after retransmitting one or more times, up to the limit specified in BACnet, and the test results are still considered a pass. The fact that retransmissions were needed should however be noted because this may be useful information for uncovering a conformance problem with the IUT.

Fail Result:

If the Result(+) response to the WriteProperty request is returned but is not correctly formed the test fails. If the WriteProperty request is confirmed but the subsequent ReadProperty indicates that the value of the property has not been correctly changed the test fails.

Inconclusive Result:

If the WriteProperty service request is confirmed but a value is not returned from the ReadProperty service because of an error, a reject, or no response is received then the test result is inconclusive. Further tests may be needed to determine if the ReadProperty or the WriteProperty service has failed. The results are also inconclusive if a reply is not received for one or more of the lower tester's requests after all retransmissions have been exhausted.

Test B.1.3.11.2 Support for the WriteProperty Service Using Default BACnet Device Name

Test Purpose:

Verify that the IUT can correctly respond to a valid WriteProperty service request with a default value for the 'BACnet Device Name' parameter in the 'Write Access Specification'.

BACnet Clause Reference: 6.5.11

Test Description:

The lower tester issues a WriteProperty service request addressed to the IUT using a default value for the 'Device Name' parameter. When the WriteProperty service request is acknowledged (possibly after retries) the lower tester issues a ReadProperty service request addressed to the IUT specifying the same object and property that was written to in the WriteProperty service request. The value returned from the ReadProperty request is then compared with the value that was written.

Expected Result:

After processing the WriteProperty request the IUT issues a Result(+) response primitive confirming that the write has been successfully completed. After processing the ReadProperty request the IUT issues a Result(+) response primitive which correctly conveys the new value for the specified object and property.

Lower Tester:

The lower tester issues a WriteProperty confirmed-request PDU with the following parameter.

Write Access Specification = (NULL, "Floor3ExhaustFan", Present_Value, ACTIVE)

After the request is issued several things can happen and the lower tester must respond as indicated below.

IUT_Response PDU	Lower_Tester_Action

none	When a timer expires retransmit as specified by BACnet
error	Test complete, no action needed
reject	Retransmit as specified by BACnet
simpleACK	Continue to the next step

At this point the lower tester issues a ReadPropery confirmed-request PDU with the following parameter.

'Read Access Specification' = ("Floor3ExhaustFan", Present_Value)

After the request is issued several things can happen and the lower tester must respond as indicated below.

IUT Response PDU	Lower Tester Action
none	When a timer expires retransmit as specified by BACnet
error	Test complete, no action needed
reject	Retransmit as specified by BACnet
ComplexACK	Evaluate response

The lower tester then restores the Present_Value of "Floor3ExhaustFan" by writing to it again with a value of INACTIVE.

IUT:

Process the confirmed-service requests as specified by BACnet.

Upper Tester:

No action required for this test case.

Pass Result:

A correctly formed Result(+) response primitive shall be returned by the IUT in response to the WriteProperty request. A correctly formed complexACK-PDU shall be returned by the IUT in response to the ReadProperty request which conveys the parameter 'Read Access Result' = ("Floor3ExhaustFan", ACTIVE).

Either or both of these results may come after retransmitting one or more times, up to the limit specified in BACnet, and the test results are still considered a pass. The fact that retransmissions were needed should however be noted because this may be useful information for uncovering a conformance problem with the IUT.

Fail Result:

If the Result(+) response to the WriteProperty request is returned but is not correctly formed the test fails. If the WriteProperty request is confirmed but the subsequent ReadProperty indicates that the value of the property has not been correctly changed the test fails.

Inconclusive Result:

If the WriteProperty service request is confirmed but a value is not returned from the ReadProperty service because of an error, a reject, or no response is received then the test result is inconclusive. Further tests may be needed to determine if the ReadProperty or the WriteProperty service has failed. The results are also inconclusive if a reply is not received for one or more of the lower tester's requests after all retransmissions have been exhausted.

Test B.1.3.11.3 Support for the WriteProperty Service Using Default Property Identifier

Test Purpose:

Verify that the IUT can correctly respond to a valid WriteProperty service request with a default value for the 'Property Identifier' parameter in the 'Write Access Specification'.

BACnet Clause Reference: 6.5.11

Test Description:

The lower tester issues a WriteProperty service request addressed to the IUT using a default value for the 'Property Identifier' parameter. When the WriteProperty service request is acknowledged (possibly after retries) the lower tester issues a ReadProperty service request addressed to the IUT specifying the same object and property that was written to in the WriteProperty service request. The value returned from the ReadProperty request is then compared with the value that was written.

Expected Result:

After processing the WriteProperty request the IUT issues a Result(+) response primitive confirming that the write has been successfully completed. After processing the ReadProperty request the IUT issues a Result(+) response primitive which correctly conveys the new value for the specified object and property.

Lower Tester:

The lower tester issues a WriteProperty confirmed-request PDU with the following parameter.

Write Access Specification = (IUT, "Floor3ExhaustFan", NULL, ACTIVE)

After the request is issued several things can happen and the lower tester must respond as indicated below.

IUT Response PDU	Lower Tester Action
none	When a timer expires retransmit as specified by BACnet
error	Test complete, no action needed
reject	Retransmit as specified by BACnet
simpleACK	Continue to the next step

At this point the lower tester issues a ReadPropery confirmed-request PDU with the following parameter.

'Read Access Specification' = ("Floor3ExhaustFan", Present_Value)

After the request is issued several things can happen and the lower tester must respond as indicated below.

IUT Response PDU	Lower Tester Action
none	When a timer expires retransmit as specified by BACnet
error	Test complete, no action needed
reject	Retransmit as specified by BACnet
ComplexACK	Evaluate response

The lower tester then restores the Present_Value of "Floor3ExhaustFan" by writing to it again with a value of INACTIVE.

IUT:

Process the confirmed-service requests as specified by BACnet.

Upper Tester:

No action required for this test case.

Pass Result:

A correctly formed Result(+) response primitive shall be returned by the IUT in response to the WriteProperty request. A correctly formed complexACK-PDU shall be returned by the IUT in response to the ReadProperty request which conveys the parameter 'Read Access Result' = ("Floor3ExhaustFan", ACTIVE).

Either or both of these results may come after retransmitting one or more times, up to the limit specified in BACnet, and the test results are still considered a pass. The fact that retransmissions were needed should however be noted because this may be useful information for uncovering a conformance problem with the IUT.

Fail Result:

If the Result(+) response to the WriteProperty request is returned but is not correctly formed the test fails. If the WriteProperty request is confirmed but the subsequent ReadProperty indicates that the value of the property has not been correctly changed the test fails.

Inconclusive Result:

If the WriteProperty service request is confirmed but a value is not returned from the ReadProperty service because of an error, a reject, or no response is received then the test result is inconclusive. Further tests may be needed to determine if the ReadProperty or the WriteProperty service has failed. The results are also inconclusive if a reply is not received for one or more of the lower tester's requests after all retransmissions have been exhausted.

Test B.1.3.11.4 Support for the WriteProperty Service Writing to a Remote Device

Test Purpose:

Verify that the IUT can correctly respond to a valid WriteProperty service request with a 'BACnet Device Identifier' parameter in the 'Write Access Specification' which specifies a remote device.

BACnet Clause Reference: 6.5.11

Test Description:

The lower tester issues a WriteProperty service request addressed to the IUT using a default value for the 'Property Identifier' parameter. The 'BACnet Device Name' parameter specifies the lower tester. This means that the IUT must issue a WriteProperty request addressed to the lower tester passing along the specifications for the object identifier, property identifier, and the new value.

Expected Result:

After processing the WriteProperty request the IUT issues a valid WriteProperty service request addressed to the lower tester specifying the same object identifier, property identifier and new value.

Lower Tester:

The lower tester issues a WriteProperty confirmed-request PDU with the following parameter.

Write Access Specification = (lower tester, "MA_SetPoint", Present_Value, 65.)

The lower tester should receive a WriteProperty request from the IUT and process it in the usual way.

note: At some point the lower tester should also receive a confirmation from the IUT that the original WriteProperty has succeeded. The draft standard is ambiguous about when this should occur. It needs to specify which of the following cases the confirmation applies to.

1) The WriteProperty request has been received.
2) The WriteProperty request has been received and a new WriteProperty request has been issued as specified in the service procedure.
3) The WriteProperty request has been received, a new WriteProperty request has been issued, and confirmation of the second request has been received.

IUT:

Process the requests as specified by BACnet.

Upper Tester:

No action required for this test case.

Pass Result:

A correctly formed WriteProperty request PDU with one of the following Write Access Specifications should be received by the lower tester.

Write Access Specification = (NULL, "MA_SetPoint", Present_Value, 65.) or
Write Access Specification = (NULL, "MA_SetPoint", NULL, 65.) or,
Write Access Specification = (lower tester, "MA_SetPoint", NULL, 65.) or
Write Access Specification = (lower tester, "MA_SetPoint", Present_Value, 65.)

The lower tester should also receive a Result(+) confirmation from the IUT which corresponds to the original WriteProperty request. The timing of this confirmation is ambiguous (see note above).

These results may come after retransmitting the original WriteProperty request one or more times, up to the limit specified in BACnet, and the test results are still considered a pass. The fact that retransmissions were needed should however be noted because this may be useful information for uncovering a conformance problem with the IUT.

Fail Result:

There are three cases which constitute a failure for this test.

1) The IUT does not issue a WriteProperty request as described above but it does issue a WriteProperty confirmation to the original request.

2) The IUT issues a correctly WriteProperty request but does not issue a WriteProperty confirmation for the original WriteProperty request.

3) The timing of the WriteProperty request and the WriteProperty confirmation is not appropriate (this case only applies after the ambiguity is resolved.)

7.1.1.4 Remote Device Management Services

7.1.1.5 Virtual Terminal Services

The purpose of Virtual Terminal services is to facilitate the bi-directional exchange of character-oriented data.

Normally, these services would be used to permit an application program in one BACnet device to act as a "terminal emulator" which interacts with an "operator interface" application program in another BACnet device. Because operator interfaces are not defined by BACnet, a test of these services must be customized for each device being tested. Guidelines for doing this will be added to a future draft of this test suite.

7.1.2 Application Services Initiated by the Implementation Under Test

The purpose of the tests in this group is to determine if the IUT can correctly initiate the application services that the PICS claims the IUT supports. The lower tester writes to the properties of the ConformTest object in the IUT to establish which application service the IUT is expected to initiate and to specify the parameters, if any, that are to be used in constructing the request PDU. The lower tester then issues a ConformTest service request which instructs the IUT to construct and issue the application service request specified by the ConformTest object.

7.1.2.1 Alarm and Event Services

B.2.1 Initiating the AcknowledgeAlarm Service

Test Purpose:

Verify that the IUT can correctly initiate an AcknowledgeAlarm service request.

BACnet Clause Reference: 6.3.1

Test Description:

The lower tester issues a series of WriteProperty requests to initialize the values of the ConformTest object. The lower tester then issues a ConformTest service request and the response from the IUT is analyzed.

Expected Result:

The IUT issues an AcknowledgeAlarm service request using the parameters specified in the ConformTest object followed by a Result(+) response to the ConformTest service request.

Lower Tester:

The lower tester issues a series of WriteProperty service requests, following the usual retry procedure if needed, to initialize the ConformTest object to the following values.

Service_Name	= AcknowledgeAlarm
Parameter1	= "Test_Alarm"
Parameter2	= ALARM
Parameter3	= current time
All other parameters	= NULL

When the confirmations to these services are received a ConformTest service request is issued.

Upper Tester:

When the ConformTest service indication is received the properties of the ConformTest object are read and an AcknowledgeAlarm service request is generated using the parameter values specified in the ConformTest object.

Pass Result:

A pass result consists of the following sequence of events in order.

1. The lower tester issues the specified WriteProperty requests and receives the corresponding Result(+) confirmations after one or more tries.

2. The lower tester issues a ConformTest service request.

3. The lower tester receives a correctly formed AcknowledgeAlarm service indication from the IUT which conveys the parameters specified in the ConformTest object.

4. The lower tester receives a correctly formed Result(+) confirmation to the previously issued ConformTest service request.

Fail Result:

If the lower tester does not receive a correctly formed AcknowledgeAlarm service indication, the test result is a failure. (note: This could be an inconclusive result if the correct implementation of the ConformTest service has not been established by its use in other test cases. It is certain however that either the ConformTest service has failed or the AcknowledgeAlarm service has failed.) The test result is also a failure if the IUT returns an error-PDU or rejects the ConformTest request and all retries.

Inconclusive Result:

The result is inconclusive if the IUT does not respond to any of the requests from the lower tester. The communication link should be checked to ensure that communication has not been lost due to a lower layer failure. If this is the case the test must be retried. If this is not the case then this test has failed.

B.2.2 Initiating the ConfirmedEventNotification Service

Test Purpose:

Verify that the IUT can correctly initiate a ConfirmedEventNotification service request.

BACnet Clause Reference: 6.3.2

Test Description:

This test case assumes that the IUT has an Event Enrollment object with the following property values.

Object Identifier	= "TestAlarm"
Event Type	= ALARM
Property Reference List	= ("MA_SetPoint", Present_Value)
Enabled	= TRUE
Notification Rules	= (NORMAL_TO_ALARM, 2, 3, TRUE, "This is a test alarm")
Parameter List	= (50, 60, 1)
Confirmed Recipient List	= lower tester
Unconfirmed Recipient List	= NULL

The event notification is triggered by writing to the Present_Value property of the "MA_SetPoint" object causing it to go out-of-range.

note: When all of the Event_Types have been defined a test similar to this one should be made for each Event_Type to check that the Event_type algorithm works properly.

Expected Result:

The IUT issues a ConfirmedEventNotification service request conveying the parameters specified in the "Pass Result" section below. The IUT also issues a Result(+) response to the WriteProperty service request.

Lower Tester:

The lower tester must establish that the "TestAlarm" object exists and is configured as indicated above. A WriteProperty service request is issued containing the following parameter.

'Write Access Specification' = ("MA_SetPoint", Present_Value, 70)

When a correctly formed ConfirmedEventNotification indication is received the lower tester must respond with a Result(+) confirmation.

IUT:

The IUT processes the WriteProperty service indication as specified by BACnet, detects that an event has occurred and issues a ConfirmedEventNotification service request.

Upper Tester:

No action is required for this test.

Pass Result:

The lower tester must receive a Result(+) confirmation to the WriteProperty service request and a ConfirmedEventNotification indication. These PDUs may arrive in either order. The ConfirmedEventNotification indication must convey the following parameters.

'Event Enrollment Object Identifier'	= "TestAlarm"
'Event State'	= ALARM
'Event Type'	= ALARM
'Priority'	= 2
'Message Test"	= "This is a test alarm"
'List of Events'	= (("MA_SetPoint", Present_Value), 70)
'Time Stamp'	= current time or a sequence number
'Acknowledgement Notification'	= FALSE

Fail Result:

If the lower tester receives a Result(+) confirmation to the WriteProperty service request but it does not receive a correctly formed ConfirmedEventNotification indication as described above the test fails.

Inconclusive Result:

If the IUT does not respond to the WriteProperty request and does not issue the ConfirmedEventNotification service request the result is inconclusive. The communication link should be checked to ensure that communication has not been lost due to a lower layer failure.

B.2.2 Initiating the UnconfirmedEventNotification Service

Test Purpose:

Verify that the IUT can correctly initiate an UnconfirmedEventNotification service request.

BACnet Clause Reference: 6.3.5

Test Description:

This test case assumes that the IUT has an Event Enrollment object with the following property values.

Object Identifier	= "TestAlarm"
Event Type	= ALARM
Property Reference List	= ("MA_SetPoint", Present_Value)
Enabled	= TRUE
Notification Rules	= (NORMAL_TO_ALARM, 2, 3, TRUE, "This is a test alarm")
Parameter List	= (50, 60, 1)
Confirmed Recipient List	= NULL
Unconfirmed Recipient List	= Lower Tester

The event notification is triggered by writing to the Present_Value property of the "MA_SetPoint" object causing it to go out-of-range.

note: When all of the Event_Types have been defined a test similar to this one should be made for each Event_Type to check that the Event_type algorithm works properly.

Expected Result:

The IUT issues an UnconfirmedEventNotification service request conveying the parameters specified in the "Pass Result" section below. The IUT also issues a Result(+) response to the WriteProperty service request.

Lower Tester:

The lower tester must establish that the "TestAlarm" object exists and is configured as indicated above. A WriteProperty service request is issued containing the following parameter.

'Write Access Specification' = ("MA_SetPoint", Present_Value, 70)

IUT:

The IUT processes the WriteProperty service indication as specified by BACnet, detects that an event has occurred, and issues an UnconfirmedEventNotification service request.

Upper Tester:

No action is required for this test.

Pass Result:

The lower tester must receive a Result(+) confirmation to the WriteProperty service request and an UnconfirmedEventNotification indication. These PDUs may arrive in either order. The UnconfirmedEventNotification indication must convey the following parameters.

'Event Enrollment Object Identifier'	= "TestAlarm"
'Event State'	= ALARM
'Event Type'	= ALARM
'Priority'	= 2
'Message Test"	= "This is a test alarm"
'List of Events'	= (("MA_SetPoint", Present_Value), 70)
'Time Stamp'	= current time or a sequence number
'Acknowledgement Notification'	= FALSE

Fail Result:

If the lower tester receives a Result(+) confirmation to the WriteProperty service request but it does not

receive a correctly formed UnconfirmedEventNotification indication as described above the test fails.

Inconclusive Result:

If the IUT does not respond to the WriteProperty request and does not issue the ConfirmedEventNotification service request the result is inconclusive. The communication link should be checked to ensure that communication has not been lost due to a lower layer failure.

7.1.2.2 File Access Services

At the present time the File Access services are in a considerable state of flux with some radically different proposals being considered by the SPC. Defining test cases for these services will be delayed until the situation becomes more clear.

7.1.2.3 Object Access Services

B.2.3.1.1 Initiating the AddListElement Service

Test Purpose:

Verify that the IUT can correctly initiate an AddListElement service request.

BACnet Clause Reference: 6.5.1

Test Description:

The lower tester issues a series of WriteProperty requests to initialize the values of the ConformTest object. The lower tester then issues a ConformTest service request and the response from the IUT is analyzed.

Expected Result:

The IUT issues a AddListElement service request using the parameters specified in the ConformTest object followed by a Result(+) response to the ConformTest service request.

Lower Tester:

The lower tester issues a series of WriteProperty service requests, following the usual retry procedure if needed, to initialize the ConformTest object to the following values.

Service_Name	= AddListElement Service
Parameter1	= Test_Group
Parameter2	= List_of_Access_Specifications
Parameter3	= (("HighPressSwitch", (Present_Value, Polarity, Reliability)), ("Floor3ExhaustFan", (Present_Value, Polarity, Reliability))
All other parameters	= NULL

When the confirmations to these services are received a ConformTest service request is issued. If the IUT issues the AddListElement service request as expected, the lower tester responds by issuing a Result(+) response primitive.

Upper Tester:

When the ConformTest service indication is received the properties of the ConformTest object are read and a AddListElement service request is generated using the parameter values specified in the ConformTest object.

Pass Result:

A pass result consists of the following sequence of events in order.

1. The lower tester issues the specified WriteProperty requests and receives the corresponding Result(+) confirmations after one or more tries.

2. The lower tester issues a ConformTest service request.

3. The lower tester receives a correctly formed AddListElement service indication from the IUT which conveys the parameters specified in the ConformTest object.

4. The lower tester issues a Result(+) response primitive.

5. The lower tester receives a correctly formed Result(+) confirmation to the previously issued ConformTest service request.

Fail Result:

If the lower tester does not receive a correctly formed AddListElement service indication the test result is a failure. (note: This could be an inconclusive result if the correct implementation of the ConformTest service has not been established by its use in other test cases. It is certain however that either the ConformTest service has failed or the AddListElement service has failed.) The test result is also a failure if the IUT returns an error-PDU or rejects the ConformTest request and all retries.

Inconclusive Result:

The result is inconclusive if the IUT does not respond to any of the requests from the lower tester. The communication link should be checked to ensure that communication has not been lost due to a lower layer failure. If this is the case the test must be retried. If this is not the case then this test has failed.

B.2.3.2.1 Initiating the CreateEnrollmentObject Service With No Optional Parameters

Test Purpose:

Verify that the IUT can correctly initiate the CreateEnrollmentObject service request with no optional parameters.

BACnet Clause Reference: 6.5.2

Test Description:

The lower tester issues a series of WriteProperty requests to initialize the values of the ConformTest object. The lower tester then issues a ConformTest service request and the response from the IUT is analyzed.

Expected Result:

The IUT issues a CreateEnrollmentObject service request using the parameters specified in the ConformTest object followed by a Result(+) response to the ConformTest service request.

Lower Tester:

The lower tester issues a series of WriteProperty service requests, following the usual retry procedure if needed, to initialize the ConformTest object to the following values.

Service_Name	= CreateEnrollmentObject
Parameter1	= "TestAlarm"
Parameter2	= ALARM

Parameter3 = ("MA_SetPoint", Present_Value)
Parameter4 = (NORMAL_TO_ALARM, 2, 3, TRUE, "This is a test alarm")
Parameter5 = (50,60,1)
All other parameters = NULL

When the confirmations to these services are received a ConformTest service request is issued. If the IUT issues the CreateEnrollmentObject service request as expected, the lower tester responds by issuing a Result(+) response primitive.

Upper Tester:

When the ConformTest service indication is received the properties of the ConformTest object are read and a CreateEnrollmentObject service request is generated using the parameter values specified in the ConformTest object.

Pass Result:

A pass result consists of the following sequence of events in order.

1. The lower tester issues the specified WriteProperty requests and receives the corresponding Result(+) confirmations after one or more tries.

2. The lower tester issues a ConformTest service request.

3. The lower tester receives a correctly formed CreateEnrollmentObject service indication from the IUT which conveys the parameters specified in the ConformTest object.

4. The lower tester issues a Result(+) response primitive.

5. The lower tester receives a correctly formed Result(+) confirmation to the previously issued ConformTest service request.

Fail Result:

If the lower tester does not receive a correctly formed CreateEnrollmentObject service indication the test result is a failure. (note: This could be an inconclusive result if the correct implementation of the ConformTest service has not been established by its use in other test cases. It is certain however that either the ConformTest service has failed or the CreateEnrollmentObject service has failed.) The test result is also a failure if the IUT returns an error-PDU or rejects the ConformTest request and all retries.

Inconclusive Result:

The result is inconclusive if the IUT does not respond to any of the requests from the lower tester. The communication link should be checked to ensure that communication has not been lost due to a lower layer failure. If this is the case the test must be retried. If this is not the case then this test has failed.

B.2.3.2.2 Initiating the CreateEnrollmentObject Service With All Optional Parameters

Test Purpose:

Verify that the IUT can correctly initiate a valid CreateEnrollmentObject service request using all optional parameters.

BACnet Clause Reference: 6.5.2

Test Description:

The lower tester issues a series of WriteProperty requests to initialize the values of the ConformTest object. The lower tester then issues a ConformTest service request and the response from the IUT is analyzed.

Expected Result:

The IUT issues a CreateEnrollmentObject service request using the parameters specified in the ConformTest object followed by a Result(+) response to the ConformTest service request.

Lower Tester:

The lower tester issues a series of WriteProperty service requests, following the usual retry procedure if needed, to initialize the ConformTest object to the following values.

Service_Name = CreateEnrollmentObject
Parameter1 = "TestAlarm"
Parameter2 = ALARM
Parameter3 = ("MA_SetPoint", Present_Value)
Parameter4 = TRUE
Parameter6 = (NORMAL_TO_ALARM, 2, 3, TRUE, "This is a test alarm")
Parameter7 = (50,60,1)
Parameter8 = GLOBAL
All other parameters = NULL

When the confirmations to these services are received a ConformTest service request is issued. If the IUT issues the CreateEnrollmentObject service request as expected, the lower tester responds by issuing a Result(+) response primitive.

Upper Tester:

When the ConformTest service indication is received the properties of the ConformTest object are read and a CreateEnrollmentObject service request is generated using the parameter values specified in the ConformTest object.

Pass Result:

A pass result consists of the following sequence of events in order.

1. The lower tester issues the specified WriteProperty requests and receives the corresponding Result(+) confirmations after one or more tries.

2. The lower tester issues a ConformTest service request.

3. The lower tester receives a correctly formed CreateEnrollmentObject service indication from the IUT which conveys the parameters specified in the ConformTest object.

4. The lower tester issues a Result(+) response primitive.

5. The lower tester receives a correctly formed Result(+) confirmation to the previously issued ConformTest service request.

Fail Result:

If the lower tester does not receive a correctly formed CreateEnrollmentObject service indication the test result is a failure. (note: This could be an inconclusive result if the correct implementation of the ConformTest service has not been established by its use in other test cases. It is certain however that either the ConformTest service has failed or the CreateEnrollmentObject service has failed.) The test result is also a failure if the IUT returns an error-PDU or rejects the ConformTest request and all retries.

Inconclusive Result:

The result is inconclusive if the IUT does not respond to any of the requests from the lower tester. The communication link should be checked to ensure that communication has not been lost due to a lower layer failure. If this is the case the test must be retried. If this is not the case then this test has failed.

B.2.3.3.1 Initiating the CreateGroupObject Service

Test Purpose:

Verify that the IUT can correctly initiate a valid CreateGroupObject service request.

BACnet Clause Reference: 6.5.3

Test Description:

The lower tester issues a series of WriteProperty requests to initialize the values of the ConformTest object. The lower tester then issues a ConformTest service request and the response from the IUT is analyzed.

Expected Result:

The IUT issues a CreateGroupObject service request using the parameters specified in the ConformTest object followed by a Result(+) response to the ConformTest service request.

Lower Tester:

The lower tester issues a series of WriteProperty service requests, following the usual retry procedure if needed, to initialize the ConformTest object to the following values.

Service_Name	= CreateGroupObject
Parameter1	= "Test_Group"
Parameter2	= (("1AH1MAT",(Min_Pres, Max_Pres, Description)), ("MA_SetPoint",(Present_Value, Units)), ("Alarm_Consoles",(Description, Device_List)))
All other parameters	= NULL

When the confirmations to these services are received a ConformTest service request is issued. If the IUT issues the CreateGroupObject service request as expected, the lower tester responds by issuing a Result(+) response primitive.

Upper Tester:

When the ConformTest service indication is received the properties of the ConformTest object are read and a CreateGroupObject service request is generated using the parameter values specified in the ConformTest object.

Pass Result:

A pass result consists of the following sequence of events in order.

1. The lower tester issues the specified WriteProperty requests and receives the corresponding Result(+) confirmations after one or more tries.

2. The lower tester issues a ConformTest service request.

3. The lower tester receives a correctly formed CreateGroupObject service indication from the IUT which conveys the parameters specified in the ConformTest object.

4. The lower tester issues a Result(+) response primitive.

5. The lower tester receives a correctly formed Result(+) confirmation to the previously issued ConformTest service request.

Fail Result:

If the lower tester does not receive a correctly formed CreateGroupObject service indication the test result is a failure. (note: This could be an inconclusive result if the correct implementation of the ConformTest service has not been established by its use in other test cases. It is certain however that either the ConformTest service has failed or the CreateGroupObject service has failed.) The test result is also a failure if the IUT returns an error-PDU or rejects the ConformTest request and all retries.

Inconclusive Result:

The result is inconclusive if the IUT does not respond to any of the requests from the lower tester. The communication link should be checked to ensure that communication has not been lost due to a lower layer failure. If this is the case the test must be retried. If this is not the case then this test has failed.

B.2.3.3.1 Initiating the DeleteObject Service

Test Purpose:

Verify that the IUT can correctly initiate a valid DeleteObject service request.

BACnet Clause Reference: 6.5.4

Test Description:

The lower tester issues a series of WriteProperty requests to initialize the values of the ConformTest object. The lower tester then issues a ConformTest service request and the response from the IUT is analyzed.

Expected Result:

The IUT issues a DeleteObject service request using the parameters specified in the ConformTest object followed by a Result(+) response to the ConformTest service request.

Lower Tester:

The lower tester issues a series of WriteProperty service requests, following the usual retry procedure if needed, to initialize the ConformTest object to the following values.

Service_Name = DeleteObject
Parameter1 = "Test_Group"
All other parameters = NULL

When the confirmations to these services are received a ConformTest service request is issued. If the IUT issues the CreateGroupObject service request as expected, the lower tester responds by issuing a Result(+) response primitive.

Upper Tester:

When the ConformTest service indication is received the properties of the ConformTest object are read and a DeleteObject service request is generated using the parameter values specified in the ConformTest object.

Pass Result:

A pass result consists of the following sequence of events in order.

1. The lower tester issues the specified WriteProperty requests and receives the corresponding Result(+) confirmations after one or more tries.

2. The lower tester issues a ConformTest service request.

3. The lower tester receives a correctly formed DeleteObject service indication from the IUT which conveys the parameters specified in the ConformTest object.

4. The lower tester issues a Result(+) response primitive.

5. The lower tester receives a correctly formed Result(+) confirmation to the previously issued ConformTest service request.

Fail Result:

If the lower tester does not receive a correctly formed DeleteGroupObject service indication the test result is a failure. (note: This could be an inconclusive result if the correct implementation of the ConformTest service has not been established by its use in other test cases. It is certain however that either the ConformTest service has failed or the DeleteObject service has failed.) The test result is also a failure if the IUT returns an error-PDU or rejects the ConformTest request and all retries.

Inconclusive Result:

The result is inconclusive if the IUT does not respond to any of the requests from the lower tester. The communication link should be checked to ensure that communication has not been lost due to a lower layer failure. If this is the case the test must be retried. If this is not the case then this test has failed.

B.2.3.6.1 Initiating the ReadMultipleProperty Service With No 'Return Read Access With Result' Parameter

Test Purpose:

Verify that the IUT can correctly initiate a valid ReadProperty service request with no 'Return Read Access With Result' parameter. All of the allowed combinations of the 'List of Read Access Specifications' are included in this test.

BACnet Clause Reference: 6.5.6

Test Description:

The lower tester issues a series of WriteProperty requests to initialize the values of the ConformTest object. The lower tester then issues a ConformTest service request and the response from the IUT is analyzed.

Expected Result:

The IUT issues a ReadMultipleProperty service request using the parameters specified in the ConformTest object followed by a Result(+) response to the ConformTest service request.

Lower Tester:

The lower tester issues a series of WriteProperty service requests, following the usual retry procedure if needed, to initialize the ConformTest object to the following values.

```
Service_Name          = ReadMultipleProperty
Parameter1            = ((lower tester, "1AH1MAT", Present_Value, Reliability, Description),
                              (NULL, "1AH1DMPR", Present_Value, Status_Flags),
                              (NULL, "HighPressSwitch", NULL))
All other parameters  = NULL
```

When the confirmations to these confirmed services are received a ConformTest service request is issued. If the IUT issues the ReadMultipleProperty service request as expected, the lower tester responds by issuing a Result(+) response primitive containing the requested property values.

Upper Tester:

When the ConformTest service indication is received the properties of the ConformTest object are read and a ReadMultipleProperty service request is generated using the specified values for the 'Read Access Specification'.

Pass Result:

A pass result consists of the following sequence of events in order.

1. The lower tester issues the specified WriteProperty requests and receives the corresponding Result(+) confirmations after one or more tries.

2. The lower tester issues a ConformTest service request.

3. The lower tester receives a correctly formed ReadProperty service indication from the IUT which specifies the 'List of Read Access Specifications' described above.

4. The lower tester issues a Result(+) response primitive containing the requested property values.

5. The lower tester receives a correctly formed Result(+) confirmation to the previously issued ConformTest service request.

Fail Result:

If the lower tester does not receive a correctly formed ReadProperty service indication the test result is a failure. (note: This could be an inconclusive result if the correct implementation of the ConformTest service has not been established by its use in other test cases. It is certain however that either the ConformTest service has failed or the ReadProperty service has failed.) The test result is also a failure if the IUT returns an error-PDU or rejects the ConformTest request and all retries.

Inconclusive Result:

The result is inconclusive if the IUT does not respond to any of the requests from the lower tester. The communication link should be checked to ensure that communication has not been lost due to a lower layer failure. If this is the case the test must be retried. If this is not the case then this test has failed.

B.2.3.6.2 Initiating the ReadMultipleProperty Service With 'Return Read Access with Result' = FALSE

Test Purpose:

Verify that the IUT can correctly initiate a valid ReadProperty service request with 'Return Read Access with Result' = FALSE. All of the allowed combinations of the 'List of Read Access Specifications' are included in this test.

BACnet Clause Reference: 6.5.6

Test Description:

The lower tester issues a series of WriteProperty requests to initialize the values of the ConformTest object. The lower tester then issues a ConformTest service request and the response from the IUT is analyzed.

Expected Result:

The IUT issues a ReadMultipleProperty service request using the parameters specified in the ConformTest object followed by a Result(+) response to the ConformTest service request.

Lower Tester:

The lower tester issues a series of WriteProperty service requests, following the usual retry procedure if needed, to initialize the ConformTest object to the following values.

```
Service_Name            = ReadMultipleProperty
Parameter1              = ((lower tester, "1AH1MAT", Present_Value, Reliability, Description),
                                (NULL, "1AH1DMPR", Present_Value, Status_Flags),
                                (NULL, "HighPressSwitch", NULL))
Return Read Access Specifications with Result = FALSE
All other parameters    = NULL
```

When the confirmations to these confirmed services are received a ConformTest service request is issued. If the IUT issues the ReadMultipleProperty service request as expected, the lower tester responds by issuing a Result(+) response primitive containing the requested property values.

Upper Tester:

When the ConformTest service indication is received the properties of the ConformTest object are read and a ReadMultipleProperty service request is generated using the specified values for the 'Read Access Specification' and the 'Return Read Access Specifications with Result' parameters.

Pass Result:

A pass result consists of the following sequence of events in order.

1. The lower tester issues the specified WriteProperty requests and receives the corresponding Result(+) confirmations after one or more tries.

2. The lower tester issues a ConformTest service request.

3. The lower tester receives a correctly formed ReadProperty service indication from the IUT which specifies the 'List of Read Access Specifications' described above and 'Return Access Specifications with Result' = FALSE.

4. The lower tester issues a Result(+) response primitive containing the requested property values.

5. The lower tester receives a correctly formed Result(+) confirmation to the previously issued ConformTest service request.

Fail Result:

If the lower tester does not receive a correctly formed ReadProperty service indication the test result is a failure. (note: This could be an inconclusive result if the correct implementation of the ConformTest service has not been established by its use in other test cases. It is certain however that either the ConformTest service has failed or the ReadProperty service has failed.) The test result is also a failure if the IUT returns an error-PDU or rejects the ConformTest request and all retries.

Inconclusive Result:

The result is inconclusive if the IUT does not respond to any of the requests from the lower tester. The communication link should be checked to ensure that communication has not been lost due to a lower layer

failure. If this is the case the test must be retried. If this is not the case then this test has failed.

B.2.3.6.3 Initiating the ReadMultipleProperty Service With 'Return Read Access with Result' = TRUE

Test Purpose:

Verify that the IUT can correctly initiate a valid ReadProperty service request with 'Return Read Access with Result' = TRUE. All of the allowed combinations of the 'List of Read Access Specifications' are included in this test.

BACnet Clause Reference: 6.5.6

Test Description:

The lower tester issues a series of WriteProperty requests to initialize the values of the ConformTest object. The lower tester then issues a ConformTest service request and the response from the IUT is analyzed.

Expected Result:

The IUT issues a ReadMultipleProperty service request using the parameters specified in the ConformTest object followed by a Result(+) response to the ConformTest service request.

Lower Tester:

The lower tester issues a series of WriteProperty service requests, following the usual retry procedure if needed, to initialize the ConformTest object to the following values.

```
Service_Name          = ReadMultipleProperty
Parameter1            = ((lower tester, "1AH1MAT", Present_Value, Reliability, Description),
                              (NULL, "1AH1DMPR", Present_Value, Status_Flags),
                              (NULL, "HighPressSwitch", NULL))
Return Read Access Specifications with Result = TRUE
All other parameters  = NULL
```

When the confirmations to these confirmed services are received a ConformTest service request is issued. If the IUT issues the ReadMultipleProperty service request as expected, the lower tester responds by issuing a Result(+) response primitive containing the requested property values.

Upper Tester:

When the ConformTest service indication is received the properties of the ConformTest object are read and a ReadMultipleProperty service request is generated using the specified values for the 'Read Access Specification' and the 'Return Read Access Specifications with Result' parameters.

Pass Result:

A pass result consists of the following sequence of events in order.

1. The lower tester issues the specified WriteProperty requests and receives the corresponding Result(+) confirmations after one or more tries.

2. The lower tester issues a ConformTest service request.

3. The lower tester receives a correctly formed ReadProperty service indication from the IUT which specifies the 'List of Read Access Specifications' described above and 'Return Access Specifications with Result' = TRUE.

4. The lower tester issues a Result(+) response primitive containing the requested property values.

5. The lower tester receives a correctly formed Result(+) confirmation to the previously issued ConformTest service request.

Fail Result:

If the lower tester does not receive a correctly formed ReadProperty service indication the test result is a failure. (note: This could be an inconclusive result if the correct implementation of the ConformTest service has not been established by its use in other test cases. It is certain however that either the ConformTest service has failed or the ReadProperty service has failed.) The test result is also a failure if the IUT returns an error-PDU or rejects the ConformTest request and all retries.

Inconclusive Result:

The result is inconclusive if the IUT does not respond to any of the requests from the lower tester. The communication link should be checked to ensure that communication has not been lost due to a lower layer failure. If this is the case the test must be retried. If this is not the case then this test has failed.

B.2.3.7.1 Initiating the ReadProperty Service With All Default Parameters

Test Purpose:

Verify that the IUT can correctly initiate a valid ReadProperty service request with default values for the 'Object Identifier' and 'Property Identifier' parameters in the 'Read Access Specification'.

BACnet Clause Reference: 6.5.7

Test Description:

The lower tester issues a series of WriteProperty requests to initialize the values of the ConformTest object. The lower tester then issues a ConformTest service request and the response from the IUT is analyzed.

Expected Result:

The IUT issues a ReadProperty service request using the parameters specified in the ConformTest object followed by a Result(+) response to the ConformTest service request.

Lower Tester:

The lower tester issues a WriteProperty service request, following the usual retry procedure if needed, to initialize the ConformTest object to the following values.

Service_Name	= ReadProperty
All other parameters	= NULL

When the confirmation to this confirmed service is received a ConformTest service request is issued. If the IUT issues the ReadProperty service request as expected, the lower tester responds by issuing a Result(+) response primitive containing the default property of the default object.

Upper Tester:

When the ConformTest service indication is received the properties of the ConformTest object are read and a ReadProperty service request is generated using all default values for the 'Read Access Specification'.

Pass Result:

A pass result consists of the following sequence of events in order.

1. The lower tester issues the specified WriteProperty request and receives the corresponding Result(+) confirmation after one or more tries.

2. The lower tester issues a ConformTest service request.

3. The lower tester receives a correctly formed ReadProperty service indication from the IUT which specifies the default property of the default object.

4. The lower tester issues a Result(+) response primitive containing the default property of the default object.

5. The lower tester receives a correctly formed Result(+) confirmation to the previously issued ConformTest service request.

Fail Result:

If the lower tester does not receive a correctly formed ReadProperty service indication the test result is a failure. (note: This could be an inconclusive result if the correct implementation of the ConformTest service has not been established by its use in other test cases. It is certain however that either the ConformTest service has failed or the ReadProperty service has failed.) The test result is also a failure if the IUT returns an error-PDU or rejects the ConformTest request and all retries.

Inconclusive Result:

The result is inconclusive if the IUT does not respond to any of the requests from the lower tester. The communication link should be checked to ensure that communication has not been lost due to a lower layer failure. If this is the case the test must be retried. If this is not the case then this test has failed.

B.2.3.7.2 Initiating the ReadProperty Service Using Default Object Identifier

Test Purpose:

Verify that the IUT can correctly initiate a valid ReadProperty service request with a default value for the 'Object Identifier' and a specified 'Property Identifier' in the 'Read Access Specification' parameter.

BACnet Clause Reference: 6.5.7

Test Description:

The lower tester issues a series of WriteProperty requests to initialize the values of the ConformTest object. The lower tester then issues a ConformTest service request and the response from the IUT is analyzed.

Expected Result:

The IUT issues a ReadProperty service request using the parameters specified in the ConformTest object followed by a Result(+) response to the ConformTest service request.

Lower Tester:

The lower tester issues a series of WriteProperty service requests, following the usual retry procedure if needed, to initialize the ConformTest object to the following values.

Service_Name	= ReadProperty
Parameter1	= (NULL, Protocol_Version)
All other parameters	= NULL

When the confirmation to these confirmed services are received a ConformTest service request is issued. If the IUT issues the ReadProperty service request as expected, the lower tester responds by issuing a Result(+) response primitive containing the Protocol_Version property of the default object. When the test has completed the lower tester issues a WriteProperty service request to return the Parameter1 property to a value of NULL (the standard configuration).

Upper Tester:

When the ConformTest service indication is received the properties of the ConformTest object are read and a ReadProperty service request is generated using the specified values for the 'Read Access Specification'.

Pass Result:

A pass result consists of the following sequence of events in order.

1. The lower tester issues the specified WriteProperty requests and receives the corresponding Result(+) confirmations after one or more tries.

2. The lower tester issues a ConformTest service request.

3. The lower tester receives a correctly formed ReadProperty service indication from the IUT which specifies the Protocol_Version property of the default object.

4. The lower tester issues a Result(+) response primitive containing the Protocol_Version property of the default object.

5. The lower tester receives a correctly formed Result(+) confirmation to the previously issued ConformTest service request.

Fail Result:

If the lower tester does not receive a correctly formed ReadProperty service indication the test result is a failure. (note: This could be an inconclusive result if the correct implementation of the ConformTest service has not been established by its use in other test cases. It is certain however that either the ConformTest service has failed or the ReadProperty service has failed.) The test result is also a failure if the IUT returns an error-PDU or rejects the ConformTest request and all retries.

Inconclusive Result:

The result is inconclusive if the IUT does not respond to any of the requests from the lower tester. The communication link should be checked to ensure that communication has not been lost due to a lower layer failure. If this is the case the test must be retried. If this is not the case then this test has failed.

B.2.3.7.3 Initiating the ReadProperty Service Using Default Property Identifier

Test Purpose:

Verify that the IUT can correctly initiate a valid ReadProperty service request using a default 'Property Identifier' in the 'Read Access Specification' parameter.

BACnet Clause Reference: 6.5.7

Test Description:

The lower tester issues a series of WriteProperty requests to initialize the values of the ConformTest object. The lower tester then issues a ConformTest service request and the response from the IUT is analyzed.

Expected Result:

The IUT issues a ReadProperty service request using the parameters specified in the ConformTest object followed by a Result(+) response to the ConformTest service request.

Lower Tester:

The lower tester issues a series of WriteProperty service requests, following the usual retry procedure if needed, to initialize the ConformTest object to the following values.

Service_Name	= ReadProperty
Parameter1	= (DEVICE object identifier, NULL)
All other parameters	= NULL

When the confirmation to these confirmed services are received a ConformTest service request is issued. If the IUT issues the ReadProperty service request as expected, the lower tester responds by issuing a Result(+) response primitive containing the System_Status property of the DEVICE object. When the test has completed the lower tester issues a WriteProperty service request to return the Parameter1 property to a value of NULL (the standard configuration).

Upper Tester:

When the ConformTest service indication is received the properties of the ConformTest object are read and a ReadProperty service request is generated using the specified values for the 'Read Access Specification'.

Pass Result:

A pass result consists of the following sequence of events in order.

1. The lower tester issues the specified WriteProperty requests and receives the corresponding Result(+) confirmations after one or more tries.

2. The lower tester issues a ConformTest service request.

3. The lower tester receives a correctly formed ReadProperty service indication from the IUT which specifies the default property of the DEVICE object.

4. The lower tester issues a Result(+) response primitive containing the System_Status property of the DEVICE object.

5. The lower tester receives a correctly formed Result(+) confirmation to the previously issued ConformTest service request.

Fail Result:

If the lower tester does not receive a correctly formed ReadProperty service indication the test result is a failure. (note: This could be an inconclusive result if the correct implementation of the ConformTest service has not been established by its use in other test cases. It is certain however that either the ConformTest service has failed or the ReadProperty service has failed.) The test result is also a failure if the IUT returns an error-PDU or rejects the ConformTest request and all retries.

Inconclusive Result:

The result is inconclusive if the IUT does not respond to any of the requests from the lower tester. The communication link should be checked to ensure that communication has not been lost due to a lower layer failure. If this is the case the test must be retried. If this is not the case then this test has failed.

B.2.3.7.4 Initiating the ReadProperty Service Using No Default Parameters

Test Purpose:

Verify that the IUT can correctly initiate a valid ReadProperty service request using no defaults in the 'Read Access Specification' parameter.

BACnet Clause Reference: 6.5.7

Test Description:

The lower tester issues a series of WriteProperty requests to initialize the values of the ConformTest object. The lower tester then issues a ConformTest service request and the response from the IUT is analyzed.

Expected Result:

The IUT issues a ReadProperty service request using the parameters specified in the ConformTest object followed by a Result(+) response to the ConformTest service request.

Lower Tester:

The lower tester issues a series of WriteProperty service requests, following the usual retry procedure if needed, to initialize the ConformTest object to the following values.

Service_Name	= ReadProperty
Parameter1	= ("1AH1MAT", State)
All other parameters	= NULL

When the confirmation to these confirmed services are received a ConformTest service request is issued. If the IUT issues the ReadProperty service request as expected, the lower tester responds by issuing a Result(+) response primitive containing the Protocol_Version property of the default object. When the test has completed the lower tester issues a WriteProperty service request to return the Parameter1 property to a value of NULL (the standard configuration).

Upper Tester:

When the ConformTest service indication is received the properties of the ConformTest object are read and a ReadProperty service request is generated using the specified values for the 'Read Access Specification'.

Pass Result:

A pass result consists of the following sequence of events in order.

1. The lower tester issues the specified WriteProperty requests and receives the corresponding Result(+) confirmations after one or more tries.

2. The lower tester issues a ConformTest service request.

3. The lower tester receives a correctly formed ReadProperty service indication from the IUT which specifies the State property of the object "1AH1MAT".

4. The lower tester issues a Result(+) response primitive containing the State property of the object "1AH1MAT".

5. The lower tester receives a correctly formed Result(+) confirmation to the previously issued ConformTest service request.

Fail Result:

If the lower tester does not receive a correctly formed ReadProperty service indication the test result is a failure. (note: This could be an inconclusive result if the correct implementation of the ConformTest service has not been established by its use in other test cases. It is certain however that either the ConformTest service has failed or the ReadProperty service has failed.) The test result is also a failure if the IUT returns an error-PDU or rejects the ConformTest request and all retries.

Inconclusive Result:

The result is inconclusive if the IUT does not respond to any of the requests from the lower tester. The communication link should be checked to ensure that communication has not been lost due to a lower layer failure. If this is the case the test must be retried. If this is not the case then this test has failed.

B.2.3.10.1 Initiating the WriteMultipleProperty Service With No 'Return Read Access With Result' Parameter

Test Purpose:

Verify that the IUT can correctly initiate a valid WriteMultipleProperty service request. All of the allowed combinations of the 'List of Write Access Specifications' are included in this test.

BACnet Clause Reference: 6.5.10

Test Description:

The lower tester issues a series of WriteProperty requests to initialize the values of the ConformTest object. The lower tester then issues a ConformTest service request and the response from the IUT is analyzed.

Expected Result:

The IUT issues a WriteMultipleProperty service request using the parameters specified in the ConformTest object followed by a Result(+) response to the ConformTest service request.

Lower Tester:

The lower tester issues a series of WriteProperty service requests, following the usual retry procedure if needed, to initialize the ConformTest object to the following values.

Service_Name	= WriteMultipleProperty
Parameter1	= (("1AH1MAT", Present_Value, Reliability, Description), (NULL, "1AH1DMPR", Present_Value, Status_Flags), (NULL, "HighPressSwitch", NULL))
All other parameters	= NULL

When the confirmations to these confirmed services are received a ConformTest service request is issued. If the IUT issues the ReadMultipleProperty service request as expected, the lower tester responds by issuing a Result(+) response primitive containing the requested property values.

Upper Tester:

When the ConformTest service indication is received the properties of the ConformTest object are read and a ReadMultipleProperty service request is generated using the specified values for the 'Read Access Specification'.

Pass Result:

A pass result consists of the following sequence of events in order.

1. The lower tester issues the specified WriteProperty requests and receives the corresponding Result(+) confirmations after one or more tries.

2. The lower tester issues a ConformTest service request.

3. The lower tester receives a correctly formed ReadProperty service indication from the IUT which specifies the 'List of Read Access Specifications' described above.

4. The lower tester issues a Result(+) response primitive containing the requested property values.

5. The lower tester receives a correctly formed Result(+) confirmation to the previously issued ConformTest service request.

Fail Result:

If the lower tester does not receive a correctly formed ReadProperty service indication the test result is a failure. (note: This could be an inconclusive result if the correct implementation of the ConformTest service has not been established by its use in other test cases. It is certain however that either the ConformTest service has failed or the ReadProperty service has failed.) The test result is also a failure if the IUT returns an error-PDU or rejects the ConformTest request and all retries.

Inconclusive Result:

The result is inconclusive if the IUT does not respond to any of the requests from the lower tester. The communication link should be checked to ensure that communication has not been lost due to a lower layer failure. If this is the case the test must be retried. If this is not the case then this test has failed.

B.2.3.11.1 Initiating the WriteProperty Service With All Default Parameters

Test Purpose:

Verify that the IUT can correctly initiate a valid WriteProperty service request with default values for the 'BACnet Device Name' and 'Property Identifier' parameters in the 'Write Access Specification'.

BACnet Clause Reference: 6.5.11

Test Description:

The lower tester issues a series of WriteProperty requests to initialize the values of the ConformTest object. The lower tester then issues a ConformTest service request and the response from the IUT is analyzed.

Expected Result:

The IUT issues a WriteProperty service request using the parameters specified in the ConformTest object followed by a Result(+) response to the ConformTest service request.

Lower Tester:

The lower tester issues a series of WriteProperty service requests, following the usual retry procedure if needed, to initialize the ConformTest object to the following values.

Service_Name	= WriteProperty
Write Access Specification	= (NULL, "ExhaustFanEnable", NULL, ACTIVE)
All other parameters	= NULL

When the confirmation to this confirmed service is received a ConformTest service request is issued. If the IUT issues the WriteProperty service request as expected, the lower tester responds by issuing a

Result(+) response primitive. When the test has completed the lower tester issues a WriteProperty service request to return the Parameter1 property to a value of NULL (the standard configuration).

Upper Tester:

When the ConformTest service indication is received the properties of the ConformTest object are read and a WriteProperty service request is generated using the specified values for the 'Write Access Specification'.

Pass Result:

A pass result consists of the following sequence of events in order.

1. The lower tester issues the specified WriteProperty requests and receives the corresponding Result(+) confirmations after one or more tries.

2. The lower tester issues a ConformTest service request.

3. The lower tester receives a correctly formed WriteProperty service indication from the IUT which conveys one of the valid 'Write Access Specification'.

4. The lower tester issues a Result(+) response primitive.

5. The lower tester receives a correctly formed Result(+) confirmation to the previously issued ConformTest service request.

Fail Result:

If the lower tester does not receive a correctly formed WriteProperty service indication the test result is a failure. (note: This could be an inconclusive result if the correct implementation of the ConformTest service has not been established by its use in other test cases. It is certain however that either the ConformTest service has failed or the ReadProperty service has failed.) The test result is also a failure if the IUT returns an error-PDU or rejects the ConformTest request and all retries.

Inconclusive Result:

The result is inconclusive if the IUT does not respond to any of the requests from the lower tester. The communication link should be checked to ensure that communication has not been lost due to a lower layer failure. If this is the case the test must be retried. If this is not the case then this test has failed.

B.2.3.11.2 Initiating the WriteProperty Service Using Default BACnet Device Name

Test Purpose:

Verify that the IUT can correctly initiate a valid WriteProperty service request with a default value for the 'BACnet Device Name' and a specified 'Property Identifier' in the 'Write Access Specification' parameter.

BACnet Clause Reference: 6.5.11

Test Description:

The lower tester issues a series of WriteProperty requests to initialize the values of the ConformTest object. The lower tester then issues a ConformTest service request and the response from the IUT is analyzed.

Expected Result:

The IUT issues a WriteProperty service request using the parameters specified in the ConformTest object followed by a Result(+) response to the ConformTest service request.

Lower Tester:

The lower tester issues a series of WriteProperty service requests, following the usual retry procedure if needed, to initialize the ConformTest object to the following values.

Service_Name	= WriteProperty
Parameter1	= (NULL, "Alarm_Consoles", Description, "This is a new Description!")
All other parameters	= NULL

When the confirmation to these confirmed services are received a ConformTest service request is issued. If the IUT issues the WriteProperty service request as expected, the lower tester responds by issuing a Result(+) response primitive. When the test has completed the lower tester issues a WriteProperty service request to return the Parameter1 property to a value of NULL (the standard configuration).

Upper Tester:

When the ConformTest service indication is received the properties of the ConformTest object are read and a WriteProperty service request is generated using the specified values for the 'Write Access Specification'.

Pass Result:

A pass result consists of the following sequence of events in order.

1. The lower tester issues the specified WriteProperty requests and receives the corresponding Result(+) confirmations after one or more tries.

2. The lower tester issues a ConformTest service request.

3. The lower tester receives a correctly formed WriteProperty service indication from the IUT which conveys one of the valid 'Write Access Specification'.

4. The lower tester issues a Result(+) response primitive.

5. The lower tester receives a correctly formed Result(+) confirmation to the previously issued ConformTest service request.

Fail Result:

If the lower tester does not receive a correctly formed WriteProperty service indication the test result is a failure. (note: This could be an inconclusive result if the correct implementation of the ConformTest service has not been established by its use in other test cases. It is certain however that either the ConformTest service has failed or the WriteProperty service has failed.) The test result is also a failure if the IUT returns an error-PDU or rejects the ConformTest request and all retries.

Inconclusive Result:

The result is inconclusive if the IUT does not respond to any of the requests from the lower tester. The communication link should be checked to ensure that communication has not been lost due to a lower layer failure. If this is the case the test must be retried. If this is not the case then this test has failed.

B.2.3.11.3 Initiating the WriteProperty Service Using Default Property Identifier

Test Purpose:

Verify that the IUT can correctly initiate a valid WriteProperty service request using a default 'Property Identifier' in the 'Write Access Specification' parameter.

BACnet Clause Reference: 6.5.11

Test Description:

The lower tester issues a series of WriteProperty requests to initialize the values of the ConformTest object. The lower tester then issues a ConformTest service request and the response from the IUT is analyzed.

Expected Result:

The IUT issues a WriteProperty service request using the parameters specified in the ConformTest object followed by a Result(+) response to the ConformTest service request.

Lower Tester:

The lower tester issues a series of WriteProperty service requests, following the usual retry procedure if needed, to initialize the ConformTest object to the following values.

Service_Name	= WriteProperty
Parameter1	= (lower tester, "MA_SetPoint", NULL, 65.)
All other parameters	= NULL

When the confirmation to these confirmed services are received a ConformTest service request is issued. If the IUT issues the WriteProperty service request as expected, the lower tester responds by issuing a Result(+) response primitive. When the test has completed the lower tester issues a WriteProperty service request to return the Parameter1 property to a value of NULL (the standard configuration).

Upper Tester:

When the ConformTest service indication is received the properties of the ConformTest object are read and a ReadProperty service request is generated using the specified values for the 'Read Access Specification'.

Pass Result:

A pass result consists of the following sequence of events in order.

1. The lower tester issues the specified WriteProperty requests and receives the corresponding Result(+) confirmations after one or more tries.

2. The lower tester issues a ConformTest service request.

3. The lower tester receives a correctly formed WriteProperty service indication from the IUT which conveys one of the valid 'Write Access Specification'.

4. The lower tester issues a Result(+) response primitive.

5. The lower tester receives a correctly formed Result(+) confirmation to the previously issued ConformTest service request.

Fail Result:

If the lower tester does not receive a correctly formed WriteProperty service indication the test result is a failure. (note: This could be an inconclusive result if the correct implementation of the ConformTest service has not been established by its use in other test cases. It is certain however that either the ConformTest service has failed or the ReadProperty service has failed.) The test result is also a failure if the IUT returns an error-PDU or rejects the ConformTest request and all retries.

Inconclusive Result:

The result is inconclusive if the IUT does not respond to any of the requests from the lower tester. The

communication link should be checked to ensure that communication has not been lost due to a lower layer failure. If this is the case the test must be retried. If this is not the case then this test has failed.

B.2.3.11.4 Initiating the WriteProperty Service Using No Default Parameters

Test Purpose:

Verify that the IUT can correctly initiate a valid WriteProperty service request using no defaults in the 'Write Access Specification' parameter.

BACnet Clause Reference: 6.5.11

Test Description:

The lower tester issues a series of WriteProperty requests to initialize the values of the ConformTest object. The lower tester then issues a ConformTest service request and the response from the IUT is analyzed.

Expected Result:

The IUT issues a ReadProperty service request using the parameters specified in the ConformTest object followed by a Result(+) response to the ConformTest service request.

Lower Tester:

The lower tester issues a series of WriteProperty service requests, following the usual retry procedure if needed, to initialize the ConformTest object to the following values.

Service_Name	= WriteProperty
Parameter1	= (lower tester, "HighPressSwitch", Out_Of_Service, TRUE)
All other parameters	= NULL

When the confirmation to these confirmed services are received a ConformTest service request is issued. If the IUT issues the WriteProperty service request as expected, the lower tester responds by issuing a Result(+) response primitive. When the test has completed the lower tester issues a WriteProperty service request to return the Parameter1 property to a value of NULL (the standard configuration).

Upper Tester:

When the ConformTest service indication is received the properties of the ConformTest object are read and a WriteProperty service request is generated using the specified values for the 'Write Access Specification'.

Pass Result:

A pass result consists of the following sequence of events in order.

1. The lower tester issues the specified WriteProperty requests and receives the corresponding Result(+) confirmations after one or more tries.

2. The lower tester issues a ConformTest service request.

3. The lower tester receives a correctly formed WriteProperty service indication from the IUT which conveys one of the valid 'Write Access Specification'.

4. The lower tester issues a Result(+) response primitive.

5. The lower tester receives a correctly formed Result(+) confirmation to the previously issued

ConformTest service request.

Fail Result:

If the lower tester does not receive a correctly formed WriteProperty service indication the test result is a failure. (note: This could be an inconclusive result if the correct implementation of the ConformTest service has not been established by its use in other test cases. It is certain however that either the ConformTest service has failed or the ReadProperty service has failed.) The test result is also a failure if the IUT returns an error-PDU or rejects the ConformTest request and all retries.

Inconclusive Result:

The result is inconclusive if the IUT does not respond to any of the requests from the lower tester. The communication link should be checked to ensure that communication has not been lost due to a lower layer failure. If this is the case the test must be retried. If this is not the case then this test has failed.

7.1.2.4 Virtual Terminal Services

The purpose of Virtual Terminal services is to facilitate the bi-directional exchange of character-oriented data. Normally, these services would be used to permit an application program in one BACnet device to act as a "terminal emulator" which interacts with an "operator interface" application program in another BACnet device. Because operator interfaces are not defined by BACnet, a test of these services must be customized for each device being tested. Guidelines for doing this will be added to a future draft of this test suite.

8. FUTURE WORK

Much work remains to be done before a conformance testing procedure is complete. Various parts of the draft standard need to be finalized. Outstanding questions regarding the minimum requirements in order to claim conformance need to be resolved, including the number of conformance classes will there will be and how they will be defined. This draft abstract test suite is a first attempt to define what the conformance tests will actually be. For most of the test cases in this suite, assumptions were made about particular object types and services are supported. Changes may need to be made if these assumptions do not appear to be reasonable when conformance classes are decided upon. Another open question is "How thorough is thorough enough?". It may be decided that some test cases described here will not be sufficient to build confidence that the IUT correctly implements the portion of the standard being tested. Other test cases may be considered overkill. The hard question of testing the IUT's responses to invalid or inopportune events has not been addressed in this first draft. All of these issues must be addressed before the abstract test suite can be finalized.

Having an abstract test suite is not the final answer to the conformance testing problem. There still needs to be a real test system that implements the tests. Only after such a system has been built and used for a period of time will there be any real confidence that the conformance tests are sound. Such a real test system is now being developed at NIST. The plan is to coordinate this activity with participants in the building industry so that prototype BACnet devices can be tested using this facility. This effort should provide valuable feedback both in terms of evaluating the BACnet standard and the conformance testing process.

REFERENCES

[1] Bushby, S. T., and Newman, H.M, "Standardizing EMCS Communication Protocols." ASHRAE Journal Vol 31, No 1, January 1989.

[2] Bushby, S. T., and Newman, H. M., "BACnet - The Emerging Communication Protocol for Building Automation Systems." ASHRAE Journal (scheduled for publication in April 1991).

[3] Fisher, D. M., "'ASHRAE Protocol': A Route to Building Systems Compatibility." Building Operating Management June, 1990 p. 56 - 62.

[4] Bushby, S. T., "Testing Conformance to Energy Management and Control System Communication Protocols - Part 1: Test Architecture." ASHRAE Transactions Vol 96, Pt 1, paper AT 90-14-3, 1990.

[5] ISO, Draft International Standard 9646-1, "Information processing systems - open systems interconnection - OSI Conformance Testing Methodology and Framework - part 1: General Concepts." International Standards Organization. Available from ANSI, 1400 Broadway, New York, NY 10018.

[6] Bushby, S. T., "Testing Conformance to Energy Management and Control System Communication Protocols - Part 2: Test Suite Generation." ASHRAE Transactions Vol 96, Pt 1, paper AT 90-14-4, 1990.

[7] ASHRAE, "Minutes of SPC 135P Application Services Working Group, September 14, 1990, San Diego, CA." SPC 135P AS-032.

[8] ISO, International Standard 7498, "Information Processing Systems - Open System Interconnection - Basic Reference Model." Available from ANSI, 1400 Broadway, New York, NY 10018.

[9] ASHRAE, "BACnet - A Data Communication Protocol for Building Automation and Control Networks." Working Draft 3. September 5, 1990.

APPENDIX 1 - ASN.1 Production for BACnetPropertyType

The comments after each line of the production indicate the status of the property with regard to functional range testing. A designation of "service" means that the property is closely linked to the service procedure for one or more of the application services other than WriteProperty or WriteMultipleProperty. The only functional range testing for these properties is part of testing the implementation of the related application service or services. A designation of "text" means that the property is a text string. The designation "enumerated" means that the property is an enumeration of several possible values that can be tested by using the WriteProperty and ReadProperty services. If there is no designation this means that the property has an undefined range in draft 3 of BACnet. An example of this is the "bias" property of the Loop object-type. It is a numeric value with no specified range.

BACnetPropertyType ::= ENUMERATED {

acknowledged	(0),	-- service
action	(1),	-- enumerated
active-text	(2),	-- text
active-vt-sessions	(3),	-- service
application-software-version	(4),	-- text
bias	(5),	
change-of-state-count	(6),	-- service
change-of-state-time	(7),	-- service
confirmed-recipient-list	(8),	
controlled-variable-reference	(9),	
datelist	(10)	
derivative-constant	(11),	
derivative-constant-units	(12),	-- text
description	(13),	-- text
device-address	(14),	
device-list	(15),	
device-type	(16),	-- text
effective-period	(17),	-- range
elapsed-active-time	(18),	
enabled	(19),	-- enumerated
event-type	(20),	-- enumerated
exception-schedule	(21),	
firmware-revision	(22),	-- text
friday-schedule	(23),	
inactive-text	(24),	-- text
instance-of	(25),	-- text
integral-constant	(26),	
integral-constant-units	(27),	-- text
list-of-access-specifications	(28),	
local-date	(29),	-- service
local-time	(30),	-- service
location	(31),	-- text
maximum-output	(32),	
max-message-length-supported	(33),	
max-pres	(34),	
max-raw	(35),	
minimum-off-time	(36),	-- range
minimum-on-time	(37),	-- range
minimum-output	(38),	

```
min-pres                          (39),
min-raw                           (40),
model-name                        (41),    -- text
monday-schedule                   (42),
notification-rules                (43),
number-of-states                  (44),
object-identifier                 (45),
object-type                       (46),    -- fixed
out-of-service                    (47),    -- enumerated
output-units                      (48),    -- text
parameter-list                    (49),
polarity                          (50),    -- enumerated
present-value                     (51),
priority-for-writing              (52),    -- range
process-units                     (53),    -- text
process-variable-reference        (54),
process-variable-value            (55),
program-change                    (56),    -- enumerated
program-state                     (57),    -- service
property-reference                (58),
property-reference-list           (59),
proportional-constant             (60),
proportional-constant-units       (61),    -- text
protocol-conformance-class        (62),    -- fixed
protocol-object-types-supported   (63),    -- fixed
protocol-services-supported       (64),    -- fixed
protocol-version                  (65),    -- fixed
reason-for-halt                   (66),    -- service
reference-object-identifier       (67),
reference-object-type             (68),    -- enumerated
reference-nickname                (69),
reliability                       (70),    -- fixed
resolution                        (71),
run-time                          (72),    -- range
saturday-schedule                 (73),
setpoint                          (74),
setpoint-reference                (75),
site-id                           (76),    -- text
state                             (77),
state0                            (78),
state1                            (79),
state2                            (80),
state3                            (81),
state4                            (82),
state5                            (83),
state6                            (84),
state7                            (85),
state-text                        (86),    -- text
status-flags                      (87),
sunday-schedule                   (88),
system-status                     (89),
tel-number                        (90),
```

```
        thursday-schedule               (91),
        time-of-reset                   (92),
        tuesday-schedule                (93),
        unconfirmed-recipient-list      (94),
        units                           (95),   -- text
        update-interval                 (96),   -- fixed
        vendor-name                     (97),   -- fixed
        vt-classes-supported            (98),   -- fixed
        wednesday-schedule              (99),
        window-size                     (100),
}
```

NIST-114A
(REV. 3-89)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY

BIBLIOGRAPHIC DATA SHEET

1. PUBLICATION OR REPORT NUMBER
NISTIR 4563

2. PERFORMING ORGANIZATION REPORT NUMBER

3. PUBLICATION DATE
APRIL 1991

4. TITLE AND SUBTITLE
A Draft Abstract Test Suite for Determining Conformance to the BACnet Protocol

5. AUTHOR(S)
Steven T. Bushby

6. PERFORMING ORGANIZATION (IF JOINT OR OTHER THAN NIST, SEE INSTRUCTIONS)
U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
GAITHERSBURG, MD 20899

7. CONTRACT/GRANT NUMBER

8. TYPE OF REPORT AND PERIOD COVERED

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (STREET, CITY, STATE, ZIP)
STRS

10. SUPPLEMENTARY NOTES

[ ] DOCUMENT DESCRIBES A COMPUTER PROGRAM; SF-185, FIPS SOFTWARE SUMMARY, IS ATTACHED.

11. ABSTRACT (A 200-WORD OR LESS FACTUAL SUMMARY OF MOST SIGNIFICANT INFORMATION. IF DOCUMENT INCLUDES A SIGNIFICANT BIBLIOGRAPHY OR LITERATURE SURVEY, MENTION IT HERE.)

The BACnet communication protocol for building automation and control systems is in an advanced state of development and is expected to be released for public review in early 1991. When the review process is completed it will become an ASHRAE standard. One of the important outstanding issues to be resolved is conformance to the standard and how to test devices to determine if they meet the conformance requirements. This report is a draft Abstract Test Suite based on working draft three of the proposed standard.

This Abstract Test Suite is a first step in developing the tests which will be used to certify conformance to BACnet. Its purpose is to provide a starting point from which a conformance certification program can be built and to focus discussions on the outstanding conformance issues that need to be resolved before the standard can be considered complete. The role of an abstract test suite in the conformance testing process is described, a proposed BACnet test system architecture is presented and individual test cases are defined. Test cases to determine support for object types and application services defined in working draft three are included. A standard object configuration is also proposed to simplify the testing process.

12. KEY WORDS (6 TO 12 ENTRIES; ALPHABETICAL ORDER; CAPITALIZE ONLY PROPER NAMES; AND SEPARATE KEY WORDS BY SEMICOLONS)
ASHRAE; BACnet; building automation; computer; conformance; control; EMCS, energy management; protocol

13. AVAILABILITY

[X] UNLIMITED

[ ] FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NATIONAL TECHNICAL INFORMATION SERVICE (NTIS).

[ ] ORDER FROM SUPERINTENDENT OF DOCUMENTS, U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, DC 20402.

[X] ORDER FROM NATIONAL TECHNICAL INFORMATION SERVICE (NTIS), SPRINGFIELD, VA 22161.

14. NUMBER OF PRINTED PAGES
118

15. PRICE
A06

ELECTRONIC FORM