BACnet® TESTING LABORATORIES

IMPLEMENTATION GUIDELINES

V 0.49

- Revised May 24, 2017

Table Of Contents

1 Introduction

The BACnet standard offers many options in the implementation of BACnet devices. Members of the BACnet Testing Labs (BTL) have agreed on specific options of capabilities that should be implemented in BACnet devices to maximize their interoperability with devices that are tested and listed by the BTL.

This document presents those options and capabilities as recommended guidelines for all BACnet implementers. It also presents guidelines to avoid mistakes made in the past by new BACnet implementers.

References to the BACnet standard, unless otherwise specified, are to *ANSI/ASHRAE Standard 135-2012, BACnet – A Data Communication Protocol for Building Automation and Control Networks*."

2 General

The following guidelines apply to all devices.

2.1 Beware of assumptions

Don't make assumptions about the behaviour of a communicating peer beyond what the BACnet standard specifies. The peer device might not implement some functionality, property or service that is designated by the standard as being optional. (Common examples of such are the basis of specific recommendations in this guide.)

In particular, be prepared to communicate with devices implemented to an earlier protocol revision. (Additions are listed in the "History of Revisions" at the end of the BACnet standard. The referenced Addenda detailing the changes are available online from ASHRAE. See **17.1**.)

2.2 Be prepared to fall back when utilizing optional features

Do not rely on other devices supporting an optional capability of BACnet; it might not be there. Whenever possible provide a fallback position by using features that are required. For example, if one wants reasonably timely data updates from a peer device and seeks to acquire them by subscribing for change of value reports (COV), fall back to polling for the data if the peer device does not support COV subscription, or if it rejects the subscription request for some other reason.

2.3 All BACnet devices shall execute ReadProperty-Request

All BACnet devices shall be able to execute the ReadProperty service request, even BACnet client devices. The BACnet standard requires this for all devices with an application layer. (This requirement may soon include BACnet routers; see note in section 3.)

2.4 A device shall execute all forms of a service

When a device is able to execute a service request, it shall be able to execute all forms of the service. This allows a client device to select the form of the service request it chooses and expect successful interoperation. "All forms of the service" include all forms of the service parameters (see 7.11 in this guide, for example), and its application to appropriate objects and services (see 7.12 in this guide).

Except as specifically noted in the standard.

It should be noted that some forms of some services will, when executed correctly by a device, always return a Result(-). For example, a device that contains no writable array properties will always return a Result(-) to WriteProperty requests that contain an Array Index parameter.

Example Exceptions:

Service	Parameter	Reason
DeviceCommunicationControl, ReinitializeDevice	'Password'	Devices may require the 'password' property to be present and therefore respond with a 'Result(-)' if this parameter is missing.
DeviceCommunicationControl	'Time Duration'	Devices which do not support an internal clock are not required to support this parameter and may return 'Result(-)' if a finite duration is specified.
SubscribeCOV	'Lifetime'	If 'IssueConfirmedNotifications' is present and 'Lifetime' is absent, a device may return a 'Result(-)' if they do not support the indefinite lifetime option.

2.5 Max_APDU_Length_Accepted might be too large for an intervening connection

The maximum size APDU that a device can generate should be either no larger than the PTP datalink's maximum APDU size of 480 octets, or the device's Max_APDU_Length_Accepted property should be configurable and this value used to restrict the size of the APDU that will be sent by the device. This will prevent messages from being dropped when they encounter intervening PTP links.[1]

An alternative recommendation is for a device, upon initiating communications with a peer device on a remote network, to determine how large an APDU can reach that device. This can be accomplished by sending service requests of different sizes to the peer device; if a response (of any kind) comes from the peer, the request reached it. If a Reject-Message-To-Network with a reject reason of 4 ("The message is too long to be routed to this DNET") is returned, the APDU is too large to reach that device. This process works even for cases where a device reports an incorrect value, such as an Ethernet to MS/TP router reporting a Max_APDU_Length_Accepted value of 1476 on its MS/TP port (see 3.10).

2.6 Polling intervals should be configurable

For devices that are capable of polling, the poll interval should be configurable and the default should be reasonable, not "as fast as possible." This reduces the risk of overloading slower LANs and swamping routers; in addition, experience shows that not all Ethernet devices might be able to handle BACnet requests sent "as fast as possible," perhaps due to internal processing delays.

2.7 Do not flood the network on startup

Upon device startup, do not flood the network with messages. This avoids overloading networks and routers, and possibly recipient devices. In particular, alarm- and COV-notifying devices (i.e. alarm of COV servers) should be designed to suppress creation of multiple notifications immediately after power-up, if possible.

2.8 For proprietary extensions, use higher numbers

When creating vendor-proprietary extensions (per Clause 23 of the standard), whether it is for proprietary object types, proprietary properties or extending enumerations, use higher-numbered values rather than assign sequentially from the lowest value allowed for vendor-proprietary extensions. This applies particularly to the BACnetEngineeringUnits enumeration, for which the reserved range may need to be extended.

~~2.9 Be prepared for large MAC addresses of at least 18 octets~~

~~IPv6 uses 16-octet addressing. Coupling this with the two-byte UDP port, even a small MS/TP device might someday receive messages with a SADR field of 18 octets, considerably longer than the largest SADR field of 7 octets mentioned in clause 6.2.2.2 of the BACnet standard.~~

~~If provision is not made for the larger MAC addresses, existing devices will not be able to issue requests to, nor reply to requests from, devices with such larger MAC addresses. (Note: the BACnet committee is discussing this issue and is looking at a way around it.)~~

2.10 Be prepared for future changes

In the future the available CHOICEs in any particular ASN.1 production (in Clause 21) might be expanded beyond what is currently defined, as might enumerations (such as BACnetPropertyIdentifier). In addition, reserved fields and values might later be defined and used.

Devices (and parsers) should be implemented to handle these situations when encountered, without crashing the device.

[1] This guideline assumes that Ethernet LANs will not be interconnected through an MS/TP LAN, with its maximum APDU size of 480 octets; and that Ethernet, ARCNET and MS/TP LANs will not be interconnected through a LonTalk LAN, with its maximum size of 206 octets. These cases present the same difficulty as joining Ethernet LANs with a PTP link, however.

2.11 Don't give up forever

If a confirmed service request fails (all APDU timeouts and retries pass without a response), don't give up forever on communicating with the remote device. If dynamic binding (Who-Is or Who-Has) is used to locate the remote device or object, fall back to occasionally initiating the Who-Is or Who-Has to locate the remote device or object.

Similarly, if a Who-Is or Who-Has intended to locate a remote device or object is sent without receiving the corresponding I-Am or I-Have, consider re-sending the Who-Is or Who-Has occasionally until the I-Am or I-Have is seen.

2.12 DeviceCommunicationControl(disable-initiation) does not require Protocol Revision 4

The BTL allows a device to support the 'disable-initiation' option for the 'enable-disable' parameter (introduced by Protocol_Revision 4) regardless of the Protocol_Revision claimed by the device.

2.13 Units properties may support values from higher Protocol Revision levels

The BTL allows a device to support values for Units property enumerations defined by Protocol_Revision levels higher than that claimed for the device.

2.14 The Accumulator and Pulse Converter objects require Protocol Revision 4

The BTL requires that devices that support the Accumulator and Pulse Converter objects must claim Protocol_Revision 4 or higher.

2.15 Be prepared to handle MAC addresses of any size less than or equal to 6 bytes

MAC addresses are not limited to 1, 2, or 6 bytes as might be thought from Table 6-2 of ASHRAE 135-2012. Any size MAC address should be allowed which is less than or equal to 6 bytes. If the device is a router, then any size MAC address with a DLEN of 7 bytes or less shall be supported.

As an example, gateways to proprietary systems are allowed to provide MAC addresses that are not included in the standard table.

2.16 Passwords shall be 20 characters or less

The standard requires that the passwords used for the DeviceCommunicationService and ReinitializeService are restricted to a maximum of 20 characters.

2.17 Addresses are expected to be no more than 6 bytes

Before Addendum 135-2008q-2 which defined the VMAC layer, the SSPC was considering the use of MAC addresses up to 18 bytes. With the adoption of 135-2008q-2 and the VMAC layer approach, addresses are expected to be no more than 6 bytes. So there is no longer a need for implementations to expect longer addresses. The support for LON addresses of length 7 is so restrictive--only usable as a DLEN and never as an SLEN--that only routers and devices that themselves will utilize the form of LON addresses of length 7, are required to support that.

2.18 Don’t restrict to only one data link type of address

There should not be any restriction in the implementation to only one data link type of address. Be prepared to communicate with devices from any data link layer, using BACnet standard addresses. Even if implementing security restricting to execute only if initiation arrives from certain addresses, allow those addresses to be any allowable BACnet addresses, from any data link.

3 The Device object

The following guidelines cover devices that contain a Device object, or that access another device's Device object.

3.1 All BACnet devices shall contain a Device object

All BACnet devices with an application layer shall contain a Device object, even BACnet clients. This is required by the BACnet standard (clause 12.10 of the BACnet standard, "There shall be exactly one Device object in each BACnet Device.").

Only BACnet routers are exempt, and only if they do not contain an application layer. However, there is currently a proposal before the BACnet committee to require routers to have application layers.

3.2 Be prepared to encounter a BACnet device without a Device object

BACnet devices should be prepared to encounter a BACnet device without a Device object, despite item. 3.1. Such devices, typically client (workstation-like) devices, have been implemented and installed.

The problem most frequently observed is a device that stalls or locks up when requests to read another device's Device object's properties fail, or are ignored.

3.3 All Device objects shall contain a non-empty Object_List property

All Device objects shall contain an Object_List property, and this property shall include the Device object. This is required by the BACnet standard.

3.4 Be prepared to read the Object_List array element by element

Some small devices that do not support segmentation have Object_List properties that are too large to transmit unsegmented. If a device needs to read another's Object_List property, be prepared to read it array element by array element.

3.5 The Device instance shall be configurable in the range of 0 to 4194302

The BTL requires the Device instance to be configurable across the entire valid range of 0 to 4194302. Device instance numbers are often used to encode information such as location (building-floor-room). Restricted configurable instance ranges hinder this practice.

Fixed, unchangeable, device instance numbers are a problem waiting to occur. Even if there would be, say, only one such device in a building, a fixed device instance can cause problems when several buildings are later joined on a campus, metropolitan or wide-area network.

The means by which the device instance can be changed is left to the implementer, whether through DIP switches, a proprietary tool or other means.

3.6 Client devices should expect to see Device instances across the entire range of 0 to 4194302

Client devices should expect to see Device instances across the entire range of 0 to 4194302. If the client device stores Device instances, it shall provide storage capability for the full instance range (22 bits).

3.7 Device instance number 4194303 is for reading a Device object's Object_Identifier

Device instance number 4194303 can be used as a "wildcard" value for reading a Device object's Object_Identifier property (to determine its Device instance). If a ReadProperty or ReadPropertyMultiple request is received for the Object_Identifier property of Device 4194303, the response shall convey the responding device's correct Device object instance.

The ability to respond to instance 4194303 as a "wildcard" value was added in Addendum 135-2001*a*; it might not be implemented in devices earlier than Protocol_Revision 4.

3.8 The length of Bit Strings might be different than expected

The length of the Device object's Bit String properties, in particular the Protocol_Services_Supported and Protocol_Object_Types_Supported properties will vary depending upon the protocol revision to which the device was implemented. Client devices should be prepared to accept Bit String values from servers with lengths longer or shorter than those defined for the Protocol_Revision value to which the client device was implemented.

The length of Protocol_Services_Supported property value

Revision	Size	Notes
0	35	
1	37	Added ReadRange(35), UTCTimeSynchronization(36),
2 – 13	40	LifeSafetyOperation(37), SubscribeCOVProperty(38), GetEventInformation(39)
14 – 17	41	WriteGroup(40)
18 – 19	44	SubscribeCOVPropertyMultiple(41), ConfirmedCOVNotificationMultiple(42), UnconfirmedCOVNotificationMultiple(43)

The length of Protocol_Object_Types_Supported property value

Revision	Size	Notes
0	18	
1	21	Added Averaging(18), Multi-state Value(19), Trend Log(20)
2	23	Added Life Safety Point(21) and Life Safety Zone(22)
3	23	
4	25	Added Accumulator(23) and Pulse Converter(24)
5	30	Added Structured View(29), and reserved bits for several other objects in review.
6	31	Added Load Control(28), Access Door(30)
7	31	Added Event Log(25) and Trend Log Multiple(27), and reserved bit for Global Group(26) object in review
8	31	
9	38	Added Access Credential(32), Access Point(33), Access Rights(34), Access User(35), Access Zone(36), Credential Data Input(37), and reserved bit for a future object type
10	51	Added BitString Value(39), CharacterString Value(40), Date Pattern Value(41), Date Value(42), DateTime Pattern Value(43), DateTime Value(44), Integer Value(45), Large Analog Value(46), OctetString Value(47), Positive Integer Value(48), Time Pattern Value(49), Time Value(50), and reserved bit for object Network Security(38) in review
11	51	Added Global Group(26) and Network Security(38)
12	51	
13	53	Added Notification Forwarder(51) and Alert Enrollment(52)
14	55	Added Channel(53) and Lighting Output(54)
15	55	
16	56	Added Binary Lighting Output(55)
17	57	Added Timer(31) and Network Port(56)
18	60	Added Elevator Group(57), Escalator(58), and Lift(59)
19	60	

3.9 The length of the BACnetLogStatus will vary depending upon the protocol revision

The length of the BACnetLogStatus in ReadRange-ACK responses, when reading the Log_Buffer property of logging objects, will vary depending upon the protocol revision to which the device was implemented.

The length of BACnetLogStatus

Revision	Size	Notes
0 – 6	2	log-disabled (0), buffer-purged (1)
7 – 19	3	Added log-interrupted (2)

3.10 The value of Max_APDU_Length_Accepted might be different on different ports

The value of Max_APDU_Length_Accepted should not exceed the maximum value defined for the datalink layer from which it is read; this could lead to attempts to send frames that are too large to be routed the device (see 2.5). For example, in an Ethernet to MS/TP router, even though a Max_APDU_Length_Accepted value of 1476 is read from its Ethernet port the value 480 should be read from its MS/TP port.

3.11 The Protocol_Services_Supported property identifies only executable services

Only bits representing services that are executable by the device shall be set to '1' in the Protocol_Services_Supported property . Bits representing services that are initiated by the device but not executed shall be set to '0'.

This property is a means for the device to advertise to other devices which services may be sent to the device with an expectation that they will be accepted and executed.

4 MS/TP

The following guidelines cover devices that participate in MS/TP LAN communications. (See clause 9 of the BACnet standard.)

4.1 Support 38.4 and 76.8k bps

Support for 9.6k and 38.4k bps is required by the BACnet standard. If at all possible, support 76.8k bps as well, for better throughput for all devices sharing the MS/TP LAN.

4.2 Support auto-bauding

All non-routing MS/TP devices should implement auto-bauding, to facilitate changing an MS/TP LAN's baud rate.

4.3 Some device types may watch the LAN to determine factors such as baud rate

Devices that route to MS/TP, or that will spontaneously initiate service requests other than I-Am (excluding devices that might initiate COV notifications but which have no subscribers), are permitted to delay start-up for a time in order to observe traffic on the LAN to determine factors such as baud rate. They are prohibited (by the BTL) from observing indefinitely; this guarantees that the MS/TP LAN will start up.

Devices that do not route to MS/TP, and that will not spontaneously initiate service requests other than I-Am when they do start up, are permitted to observe traffic until they have sufficient information to complete their initialization and start.

4.4 MS/TP MAC addresses should be configurable

MAC addresses of MS/TP devices should be configurable by some means, whether by DIP switches or a configuration tool. Support for the entire address range (0..127 for masters and 0..254 for slaves) is required by the BACnet standard.

5 Segmentation

The following guidelines cover devices that support segmentation (see clause 5.3, BACnet-2001).

It should be noted that there exists an implicit assumption: If a device supports segmentation for both transmit and receive, the maximum number of segments it can send and receive are identical.

5.1 Support all services with all segments full

If the device supports segmentation and advertises a maximum number of supported segments, the BTL requires that it shall be able to execute any service request (that could be large enough to require segmentation) with all segments filled.

This rule applies in general to service initiation as well, though there may be exceptions such as restricting the number of property requests (and the datatypes requested) contained in a ReadPropertyMultiple request so that the response, including worst-case error returns, will not overrun the number of supported segments.

5.2 The maximum number of segments that can be accepted is advertised in two places

The maximum number of segments that can be accepted by a device is "advertised" in two places: the Max_Segments_Accepted property of the Device object (clause 12.10.19 of BACnet-2001) and in the 'max-segments-accepted' parameter of the BACnet-Confirmed-Request-APDU (See clause 20.1.2.4). This value must be at least 2.

The value of the Max_Segments_Accepted property, if present, of a peer device should be considered when initiating a service request to that peer. Devices responding to a service request shall use the 'max-segments-accepted' parameter of a confirmed service request to determine if the response is transmittable (See clause 5.4.5.3, transition CannotSend SegmentedComplex-ACK.).

5.3 Indicate the current max-segments-accepted limitation in the request APDU

If the number of the segments that can be received in the response for a particular request is less than that displayed in the Max_Segments_Accepted property of the Device object for some reason, such as other concurrent requests using up available buffers, a smaller value shall be indicated in the 'max-segments-accepted' parameter of the BACnet-Confirmed-Request-APDU.

The value of zero for this parameter, indicating "unspecified number of segments accepted," is present for backwards compatibility with older devices, prior to Addendum 135-1995*e*, and should not be used (see 5.5).

5.4 If Max_Segments_Accepted is not present, the maximum number of segments might be two

If a peer device does not support the Max_Segments_Accepted property of its Device object but indicates that it supports segmentation, the maximum number of segments of a message that the peer device will accept might be as low as two. The Max_Segments_Accepted property was introduced in protocol revision 2 of the standard.

5.5 If 'max-segmented-accepted' is '000', the maximum number of segments might be two

If the 'max-segments-accepted' parameter of a confirmed service request is '000', indicating "unspecified number of segments accepted") and the 'segmented-response-accepted' parameter is TRUE, the maximum number of segments of a message that the peer device will accept might be as low as two. The use of B'000' signifying that the initiating client itself does not know the maximum number of segments which it can accept, should be avoided since that makes it impossible for the server to determine if the response is transmittable.

5.6 Devices supporting segmentation shall be able to use non-segmented methods to retrieve large amounts of data

Client devices that support segmentation and retrieve large amounts of data shall be prepared to use non-segmented services when server devices indicate they do not support segmentation.

For example, the Object_List for a device may not fit into a single segment. In this case, the individual array items shall be read. One method is to read the size of the Object_List property and then read each item individually using the ReadProperty service or a group of items could be read using ReadPropertyMultiple service provided that each request and answer response can be placed in a single segment.

5.7 Devices shall be prepared to interoperate with implementations that claim segmentation in one direction only

Devices shall be able to interoperate with implementations that support segmentation on transmit only. Devices shall be able to interoperate with implementations that support segmentation on receive only.

6 Device Binding

The following guidelines cover the implementation of the Who-Is, I-Am, Who-Has and I-Have service requests, which are used to locate ("bind to") specific devices.

6.1 Implement I-Am and I-Have

The Who-Is service is a convenient mechanism used by most devices to bind to peer devices. A few devices use the Who-Has request instead. For this reason all devices should execute the Who-Is and Who-Has requests and initiate, in response, the I-Am or I-Have request. MS/TP slave devices are exempt because they cannot initiate requests, however there are devices that cannot communicate with the slave devices. See 6.8.

It is preferable, but not required, that I-Am and I-Have requests generated as a result of a Who-Is or Who-Has request be broadcast only on the network where the Who-Is originated. This reduces the amount of unnecessary network traffic elsewhere in the system.

6.2 Broadcast an I-Am on start-up

It is a common practice to broadcast an I-Am globally (to the entire internetwork) on start-up. Though this practice can slow startup of large systems after a power failure, it allows client devices that have already started to bind to this device sooner, and may restore normal operation more quickly when a device has been moved or replaced, perhaps with a device with a different MAC address.

6.3 Use Who-Is and I-Am for device binding

Use Who-Is and I-Am for locating devices on the internetwork ("device binding").

When a device with a worldwide unique MAC address (such as an Ethernet MAC address) is replaced, static binding (entering a device's network number and MAC address) can create a lot of error-prone work, because every single static reference to the device that was replaced, throughout the entire system, needs to be reconfigured.

6.4 Do not periodically broadcast I-Am and I-Have

Although the BACnet standard allows I-Am and I-Have requests to be initiated at any time by a BACnet device, these requests should not be periodically broadcast. This creates unnecessary traffic that can impede PTP connections and slower LANs. In addition, some devices undertake additional actions (in the form of confirmed service requests to the device that sent the I-Am) in response to an I-Am because the device may have been replaced, the device may have been replaced with a different type of device, or its location may have changed.

6.5 Restrict the Who-Is device instance range

Do not issue globally broadcast Who-Is messages (i.e., without 'Device Instance Range Low/High Limit' parameters), except as the first step of mapping a system such as by a workstation. In large systems these cause massive numbers of I-Am broadcasts, causing I-Am responses and other messages to be lost. It is better to issue a single Who-Is request for each specific peer device that is to be located.

6.6 Space out Who-Is broadcasts

Do not issue a number of Who-Is requests, one immediately after the next. The resulting flood of Who-Is and I-am messages can overwhelm routers and slower connections and LANs. It is better to space the requests out, say, one per second or one per 100 mS, rather than issue them all at once.

6.7 Reduce the rate of repeated Who-Is broadcasts

If it is not absolutely essential that a device's function requires the earliest possible establishment of communications with a peer (for example, if it is not involved in life- or equipment-safety operations), the rate at which an unanswered Who-Is for that peer is repeated should be reduced. If part of a system becomes unreachable, say by a router going offline, the resulting flood of rapidly-repeated Who-Is requests for devices on the other side of that router can impede PTP communication and slower LANs.

Where the peer-to-peer communications are less than critical, it is better to initially repeat the unanswered Who-Is every 10 seconds, or the time specified by the Device object's APDU_Timeout property, increasing over time to some maximum interval, say, 5 minutes or the time calculated by multiplying the value of the Device object's APDU_Timeout property by (1 plus the value of APDU_Retries).

6.8 Client devices should support static binding

Client devices should be able to be configured with the network number and MAC address of a device with which they will be communicating, as an alternative to the device instance. This will allow them to communicate with MS/TP slaves or other devices that do not support Who-Is and Who-Has execution.

6.9 Support for proxy I-Am responses for MS/TP devices does not require Protocol Revision 4

The BTL allows a device to support proxy I-Am responses for MS/TP devices regardless of the Protocol_Revision claimed by the device. Support for the feature is indicated by the presence of the Slave_Proxy_Enable property.

7 Data Sharing

The following guidelines apply to all devices with an application layer, i.e., they acquire data from or provide data to other devices in BACnet APDUs.

7.1 Support ReadPropertyMultiple

Support ReadPropertyMultiple execution (and, if a client device, ReadPropertyMultiple initiation) even in small devices. The improvement in network bandwidth when data is being polled is significant.

7.2 Clients should be able to fall back to smaller ReadPropertyMultiple requests

Devices that initiate ReadPropertyMultiple requests should be able to retry with, or be reconfigured to use, ReadPropertyMultiple requests for fewer property values if there is some indication that the data cannot be conveyed to the client. This will help ensure interoperation between the two devices.

The indicators received from the peer device typically are an Abort (BUFFER_OVERFLOW) APDU or an Abort (SEGMENTATION_NOT_SUPPORTED) APDU (see clause 5.4.5.3 in the BACnet standard, transition 'CannotSendSegmented-ComplexACK'). Another indicator could be that a reply is never received, if there is an intervening connection (LonTalk or MS/TP LAN, or PTP connection) unable to convey the reply. (See item 2.5.)

7.3 Clients should be able to fall back to ReadProperty requests

Devices that initiate ReadPropertyMultiple requests should be able to revert to ReadProperty requests if the server device does not indicate (on the server's Device object's Protocol_Services_Supported property) support ReadPropertyMultiple execution , or (as in 7.2) if there is some indication that the data cannot be conveyed to the client, or if an Error PDU conveying the code SERVICE_NOT_SUPPORTED or a Reject PDU conveying the code UNRECOGNIZED_SERVICE is received. This will help ensure interoperation between the two devices.

7.4 Do not poll as fast as the network allows, and polling intervals should be configurable

A device that polls other devices for data should not poll as fast as the network will allow; this can impede PTP communication and slower LANs. The interval between polls should also be configurable.

7.5 If multiple devices are being polled, alternate the polls

A device that is polling multiple peer devices for data at the same rates should, if possible, rotate the requests among the peer devices "round-robin" rather than send a number of ReadProperty (or ReadPropertyMultiple) requests to one device, then a number of requests to the next device, and so on.

7.6 Do not subscribe for COV notifications with 'Lifetime' set to zero (indefinite lifetime)

Devices should not issue SubscribeCOV or SubscribeCOVProperty requests with the 'Lifetime' parameter set to zero. (The value zero is prohibited for SubscribeCOVProperty requests.) The COV subscription list is not guaranteed to remain through a device reset or power loss, or the device holding the subscription could fail and be replaced. Worse, if the subscription list is permanently held and the subscribing device is removed, the storage for the removed device's subscription remains allocated forever.

SubscribeCOV requests should be issued with a non-zero 'Lifetime' parameter for some period (such as the amount of time it is acceptable for the subscribing device to operate with out-of-date data), and the subscription should be refreshed periodically.

7.7 If a COV subscription fails, poll for data and/or inform the operator

COV subscriptions might not always succeed, even with devices that execute SubscribeCOV or Subscribe-COVProperty requests. Devices are permitted to support a limited number of subscriptions, for example (clauses K.1.12 and K.1.13 in the BACnet standard require only five concurrent subscriptions). If possible, some action to correct the situation should be initiated.

One possible course of action is to fall back to polling for the data. Another (not necessarily exclusive) is to notify the operator of the failure to subscribe so that a change can be made in the system configuration.

7.8 COV notifications from proprietary objects should include Present_Value and Status_Flags

COV notifications from proprietary objects should include the values of the Present_Value and Status_Flags properties in addition to any other property values included in the notifications. This is for consistency with most other COV notifications.

7.9 List properties shall be modifiable using WriteProperty

List properties that may be modified using BACnet services shall also be writable using the WriteProperty service request (and WritePropertyMultiple, if supported). Modifications of such properties shall not be restricted (by the server device) to the AddListElement and RemoveListElement service requests, for better interoperability.

7.10 List properties should be modified using AddListElement and RemoveListElement

In order to reduce the potential for conflict when two or more clients are modifying a list property concurrently, the list should be modified using AddListElement and RemoveListElement service requests. (However, if an element of a list is removed by one RemoveListElement request and a subsequent RemoveListElement request specifies that element, the latter request will fail in its entirety.)

7.11 ReadPropertyMultiple execution shall support ALL, REQUIRED, and OPTIONAL

The BACnet standard requires that ReadPropertyMultiple service execution supports the special property identifiers ALL, REQUIRED, and OPTIONAL. (See clause 15.7.2 in the BACnet standard.) Some workstation devices rely on these special properties for accessing other devices' objects. If no optional properties are supported by the referenced object when reading with the OPTIONAL property, then the result shall be an empty 'List of Results' (in the presence of no other errors). If a property which is not readable using the ReadPropertyMultiple service is in the specified object and would match the special property identifier, then the entry for that property shall contain 'Error Class': PROPERTY and 'Error Code': READ_ACCESS_DENIED. The Property_List property, if present in the object, shall not be returned to ReadPropertyMultiple service requests specifying the special property identifiers ALL and REQUIRED.

7.12 ReadRange execution shall support all list properties

The BACnet standard requires that ReadRange service execution support any property that is a list. ReadRange service execution is not allowed to be restricted to logging objects' Log_Buffer properties, if there are other List properties in the device.

7.13 Inter-related properties should be read and written together for consistency in their values

Inter-related properties, such as High_Limit and Low_Limit should be read and written together using ReadPropertyMultiple and WritePropertyMultiple. Although this does not guarantee consistency between the values, because the operations are not required to be atomic, it may improve the likelihood of consistency over access using ReadProperty and WriteProperty services.

7.14 Writable Character String properties should support strings of useful length

Writable Character String properties should support strings of useful length. Although "useful" is a vague term, an Object_Name property that can contain at most a four octet Character String is not likely to be useful to a user, whereas an Object_Name of length 512 is likely to be far more than adequate.

7.15 Client devices should be able to handle Signed and INTEGER values up to 32 bits

Client devices should be able to handle Signed and INTEGER values at least up to 32 bits in length, unless the particular value is specifically restricted by the BACnet standard (such as Unsigned8). This is the minimum recommended by the BTL for interoperability with other devices.

7.16 Client devices should be able to handle Enumerated values up to 16 bits

Client devices should be able to handle Enumerated values at least up to 16 bits in length, unless the particular value is specifically restricted by the BACnet standard. This is the minimum recommended by the BTL for interoperability with other devices.

7.17 Time and Date wildcard values should only be used for unspecified time or unspecified date or in patterns

Time and Date wildcard values (X'FF') as standard clauses 20.1.12 and 20.1.13 specify, should only be used in service requests and property values as specified in the standard for unspecified time or unspecified date, when all elements of the Time or Date value are unknown, or when the standard specifies the use of time pattern, date pattern or datetime pattern. Time and Date values should not use wildcard values when conveying actual time information, even for the Day of Week or Hundredths of Seconds. Use of Date wildcard values (X'FF') as patterns in any dateRange, including in dateRange calendarEntry elements, is specifically forbidden.

7.18 Workstation devices should be able to read all basic datatypes

Operator workstations should be able to read all basic ("application," or "primitive") datatypes (as enumerated in clause 20.2.1.4 of the BACnet standard), and be able to read basic datatypes from proprietary properties. This is recommended for improved interoperability with other devices.

7.19 Support as-yet-undefined and proprietary object properties with primitive datatypes

Devices that can read and write properties in other devices should be configurable to read and write properties in as-yet-undefined standard objects and proprietary objects, as well as as-yet-undefined standard properties and proprietary properties in any objects, at least where those properties have primitive datatypes.

This improves the ability of such devices to access new objects as they are defined, as well as access to other manufacturers' proprietary objects.

7.20 Use detailed error reporting when all ReadPropertyMultiple accesses fail

The BTL recommends that if all of a ReadPropertyMultiple request's accesses fail, the response should be a 'Result(+)' primitive returning an error class and code for each access rather than a 'Result(-)' primitive conveying a single error code and class, particularly where the error classes and codes differ for different accesses.

7.21 ReadPropertyMultiple data values are returned in the order requested

The data values conveyed in a ReadPropertyMultiple-ACK shall be returned in the order in which they are requested in the ReadPropertyMultiple-Request. This is required by Clause 15.7.2 of the BACnet standard.

7.22 Support NaN and +/−INF values for REAL and Double datatypes

IEEE REAL and Double numbers have encodings for NaN (Not a Number) as well as plus and minus infinity. If such a value is written to the device, an error reply may be appropriate. Be prepared to process such a value "appropriately" if it is returned in a response to a request such as ReadProperty.

7.23 The absence of a Priority_Array property does not indicate the Present_Value property is read only

The Present_Value property may be writable even if the value object does not support the Priority_Array property. If the property is writable but not commandable, the device shall ignore the priority sent on a WriteProperty, WriteGroup or WritePropertyMultiple command.

7.24 Server devices are required to support all COV lifetime values up to 8 hours (28800 seconds)

The BTL requires that devices that execute SubscribeCOV or SubscribeCOVProperty (COV servers) accept all lifetime values in the range 1 through 28800 seconds (8 hours). COV server devices may support values outside the range as well.

The BTL requires support for this range of lifetime values in order to ensure that COV clients are able to select lifetime values that will not be rejected by COV server devices. This was clarified in protocol revision 8 of the standard (see Addendum 135-2008*s*).

7.25 Client devices are required to support a COV lifetime value within 1 to 28800 seconds (up to 8 hours)

The BTL requires that COV clients be able to subscribe with a lifetime in the range 1 through 28800 seconds. Client devices need not support all values in the range, nor are they restricted to supporting values only in that range.

The BTL requires that COV clients support a value within this range to ensure that COV clients can subscribe with a lifetime value that will be accepted by all COV servers. This was clarified in protocol revision 8 of the standard (see Addendum 135-2008*s*).

7.26 Input object's Present_Value should be read-only

While the standard does not prohibit this functionality it is the position of the BTL-WG that when an input object's Out_Of_Service property is FALSE, the Present_Value should be read-only.

7.27 INTEGER datatype encoding differs from Unsigned encoding

INTEGER datatype encoding differs in its data from Unsigned encoding, by the existence of a leading 0 octet on positive values whose high bit in the second octet is set. A common programming mistake of omitting that leading 0 octet, for example the value 180, would make the value instead equal -76.

8 Arrays

The following guidelines cover the implementation of arrays, including the priority arrays found in "Output" and "Value" object types (such as Analog Output or Multi-state Value), and described in clause 19.2 of the BACnet standard.

8.1 Support for array element 0 is required

Support for array element 0, which reports the size of the array, is required by the BACnet standard for all arrays, including the Device object's Object_List property. (See clause 12 of the BACnet standard.) In the latter case, devices sometime read element 0 to determine how they will proceed to read the Object_List property (see 3.4). The BACnet standard defines the datatype of array element 0 to be Unsigned for all arrays, regardless of the datatype of the other elements.

8.2 Arrays shall be readable in their entirety (except…)

The BTL requires that array properties shall be readable in their entirety, such as with the ReadProperty service, except when the response is too large to fit in a single APDU and either the client or the server device does not support segmentation. The BTL does not allow devices to arbitrarily restrict read access methods, in order to improve interoperability.

8.3 Resizable arrays should be resizable per Addendum 135-2001*a*-8

Resizable arrays should be resizable by the methods specified in Addendum 135-2001*a*-8 for improved interoperability (in BACnet-2004 the method is described in the initial paragraphs of Clause 12).

8.4 Client devices should be prepared for array indexes of at least 16 bits

Client devices should be prepared to handle array indexes of at least 16 bits; this includes the size of array element 0, which reports the size of the array. This is recommended for interoperability with other devices.

8.5 All 16 priority array levels shall be present

When implementing priority arrays, all 16 levels shall be present. The BACnet standard does not allow an implementation to have fewer levels.

8.6 Priority array level 6 is not writable

The BACnet standard reserves Level 6 of the priority array for the Minimum On/Off Time algorithm and prohibits its use for any other purpose in any object. (Clause 19.2.3 of the BACnet standard: "Command priority 6 is reserved for use by this algorithm and may not be used for other purposes in any object.") As such, writes using BACnet services should result in 'Result(-)' responses.

8.7 If the device cannot afford a priority array in an "Output" object, use a "Value" object

All output objects shall have priority arrays, but value objects are not required to have them. If the device needs to have an object that represents a physical output but cannot afford a priority array, use the corresponding "Value" object type instead (for example, a Binary Value object instead of a Binary Output).

8.8 Re-evaluate priority array as soon as possible upon write

Priority arrays should be evaluated to update the Present_Value property as soon as possible when written, preferably before the write is ACKed, but in particular before a read request for the Present_Value or Priority_Array properties is serviced. Extensively delayed updates can confuse operators and can cause problems in testing.

9 Alarms

The following guidelines apply to devices that initiate or receive alarms, i.e. ConfirmedEventNotification or UnconfirmedEventNotification messages.

9.1 Use the standard event algorithms if at all possible

When implementing event initiation, use the standard event algorithms if at all possible. There are at present devices that are not able to parse proprietary 'Event Values' (the event notification parameters). (See 9.2.)

9.2 Parse all event notifications

Devices that receive and process event notifications should be able to receive and process proprietary event types, where the construction of the 'Event Values' (the notification parameters) is not known. The parser should be able to parse through this field, even though the values will be discarded, in order that the notification can be received and processed. A device should not drop (ignore) event notifications simply because it does not know the event type of the notification; this is especially important for devices that support the AE-N-A BIBB.

Devices that at present are unable to parse unknown event type parameters should be upgraded to do so.

9.3 Do not implement the Event Enrollment object's Recipient property

Do not implement the Recipient property of Event Enrollment objects. This property was removed in Addendum 135-2001*a*-4. Use a Notification_Class object instead.

9.4 Support all forms of BACnetRecipient and the DM-DDB-A BIBB

The BTL requires that for devices that generate alarms or events (i.e. support the AE-N-I-B, AE-N-E-B, or T-ATR-B BIBBs) to receive a BTL Listing, all forms of the 'Recipient' parameter of BACnetDestination elements of the Recipient_List property of the Notification Class object shall be supported. In addition to this the BTL requires support for the DM-DDB-A BIBB (initiate the Who-Is service in order to locate alarm recipients).

9.5 Do not rely on the GetAlarmSummary service for acknowledging unseen alarms

GetAlarmSummary does not provide sufficient information to acknowledge alarms not seen by the acknowledging device. Devices that need to acknowledge outstanding alarms should be able to initiate GetEventInformation.

9.6 Support the GetEventInformation service

The BTL requires that for devices that generate alarms or events (i.e. support the AE-N-I-B or AE-N-E-B BIBBs) to receive a BTL Listing, the devices shall be able to execute the GetEventInformation service.

9.7 GetEventInformation requires APDU sizes greater than 50 or segmentation

Devices that initiate the GetEventInformation service (AE-INFO-A) need to be able to receive APDUs larger than 50 octets, or be able to receive segmented messages, because the GetEventInformation-ACK (response) will not fit in a 50-octet APDU.

Likewise, devices that execute the GetEventInformation service (AE-INFO-B) need to be able to send APDUs larger than 50 octets, or be able to transmit segmented messages.

9.8 The Notification Class object's Recipient_List property should be persistent

The contents of the Notification Class object's Recipient_List property should not be lost when the device is powered down or reset. There is no standard mechanism for restoring this information from some other source when the device starts up.

9.9 B-AWS devices should be able to modify standard alarm properties of standard objects

Operator workstations claiming the B-AWS device profile should be able to modify standard alarm properties of standard objects in order to change alarm configurations. These properties are:

Alarm_Value	Event_Enable	High_Limit	Notification_Class
Alarm_Values	Event_Parameters	Life_Safety_Alarm_Values	Notify_Type
Deadband	Fault_Values	Limit_Enable	Time_Delay
Error_Limit	Feedback_Value	Low_Limit	

Workstations should also be able to modify the Event_Type property of devices implemented prior to Addendum 135-2001*c*-3.

9.10 Unless fixed by the application, standard alarm parameters should be writable

Unless fixed by the device's application, the standard properties presenting alarm parameters should be writable to allow modification by workstations. These properties are:

Alarm_Value	Event_Enable	High_Limit	Notification_Class
Alarm_Values	Event_Parameters	Life_Safety_Alarm_Values	Notify_Type
Deadband	Fault_Values	Limit_Enable	Time_Delay
Error_Limit	Feedback_Value	Low_Limit	

9.11 Recipient_List shall be writable

The BTL requires that for devices that claim a Device profile that requires generation of alarms or events to receive a BTL Listing, the Recipient_List property in Notification Class and any Notification Forwarder objects shall be writable. If a device claims a Device profile that does not require alarming, it can make use of the read-only Notification Class Recipient_List property value with "Recipient set to a local broadcast, Valid Days set to all days, From Time set to 00:00:00.0, To_Time set to 23:59:59.99, Process Identifier set to 0, Issue Confirmed Notifications set to FALSE, and all bits in Transitions set to TRUE" which is designed to distribute events via Local Broadcast to external Notification Forwarder objects.

9.12 Support both ConfirmedEventNotification and UnconfirmedEventNotification

The BTL requires that for a device that generates alarms or events to receive a BTL Listing, the device shall be able to initiate both ConfirmedEventNotification and UnconfirmedEventNotification service requests.

Likewise, the BTL requires that for a device receives alarms or events to receive a BTL Listing, the device shall be able to execute both ConfirmedEventNotification and UnconfirmedEventNotification service requests.

9.13 UnconfirmedEventNotification Execution requires APDU sizes greater than 50

Devices that execute the UnconfirmedEventNotification service (AE-N-A) shall be able to receive APDUs larger than or equal to 128 octets, because many of the event types conveyed by an UnconfirmedEventNotification request will not fit in a 50-octet APDU.

9.14 Accept any Acknowledging Process Identifier in the AcknowledgeAlarm service request

The BTL requires that a device (regardless of protocol revision) accept any Acknowledging Process Identifier parameter in the AcknowledgeAlarm service. This requirement was added to the standard in Protocol_Revision 7.

Some implementations have required that the Acknowledging Process Identifier match a configured recipient of an alarm for the AcknowledgeAlarm service request to succeed. This prevents operator workstations that learn of an alarm by means other than receiving an alarm from acknowledging the alarm.

9.15 Life Safety objects may advertise supported modes despite Protocol_Revision

The BTL permits devices claiming a Protocol_Revision less than 4 to advertise supported life safety modes in the Life Safety Point and Life Safety Zone object's Accepted_Modes property (introduced in Addendum 135-2001*c*-1).

9.16 LifeSafetyOperation execution Unsilence operations require Protocol_Revision 4

The BTL requires that devices in which the Unsilence operations (unsilence, unsilence-audible, unsilence-visual) support execution of the 'request' parameter of the LifeSafetyOperation service request (introduced in Addendum 135-2001*c*-2) to claim Protocol_Revision 4 or higher.

9.17 The Event_Parameters property sets the event type, the Event_Type property indicates it

Addendum 135-2001*c*-3 (Protocol_Revision 4) made it clear that the event type of the Event Enrollment object is set by writing its Event_Parameters property and that the Event_Type property indicates the event type. Devices at all Protocol_Revisions should implement this relationship between the two properties.

9.18 Use the event priorities found in Annex M

The BTL suggests that the event priorities defined in Annex M (introduced in Addendum 135-2001c-6) be implemented, independent of the Protocol_Revision claimed by the device.

9.19 The Life Safety object behavior in Addendum 135-2004*a*-1 requires Protocol_Revision 5

The BTL requires that devices supporting Life Safety Point and Life Safety Zone objects, and implementing the behaviors described in Addendum 135-2004*a*-1, shall claim Protocol_Revision 5.

9.20 To indicate “all day” in BACnetDestination use 00:00:00.00 to 23:59:59.99

For client devices configuring a BACnetDestination, indicate that a particular destination is viable all day by using 00:00:00.00 for ‘From Time’ and 23:59:59.99 for ‘To Time’. The intent here is unambiguous, unlike the use of 00:00:00.00 for ‘To Time’ which can be interpreted as "active only at 00:00:00.00."

9.21 If the intrinsic alarming properties are present in an object, it shall support intrinsic alarming

If the properties which support intrinsic alarming are present in an object, it shall support intrinsic alarming. If the object supports intrinsic alarming solely for FAULT reporting, the properties which support intrinsic alarming are nonetheless all present, including vestigial properties such as Limit_Enable and Time_Delay which are not operative for FAULT reporting.

9.22 Rapid transitions in the OUT_OF_RANGE and FLOATING_LIMIT algorithms

The behavior of the OUT_OF_RANGE and FLOATING_LIMIT when the Present_Value transitions faster than Time_Delay between the ranges associated with the High_Limit and Low_Limit states was clarified by the BACnet Committee in Interpretation 135-2004-8 and changes to the standard in Addendum 135-2008*r*.

"An object that reports FLOATING_LIMIT or OUT_OF_RANGE events shall not generate OFFNORMAL transitions from an OFFNORMAL (LOW_LIMIT or HIGH_LIMIT) state."

That behavior was further changed in the standard in Addendum 135-2012*af*, however, to optionally allow implementing this behavior. See clauses 13.3.5 and 13.3.6

When the Present_Value transitions between the ranges associated with the High_Limit and Low_Limit states and the Time_Delay_Normal associated with transitions to NORMAL,is less than the Time_Delay associated with transitions to OFFNORMAL, it was clarified by the BACnet Committee in Interpretation 135-2016-2 that FLOATING_LIMIT or OUT_OF_RANGE events shall be generated for NORMAL transitions at the moment when Time_Delay_Normal is reached. Then the event shall be generated for the OFFNORMAL transition when Time_Delay has elapsed.

10 Scheduling

The following guidelines apply to devices that contain Calendar and Schedule objects, and to workstations that can modify those objects.

10.1 Schedules should be modifiable

Unless the application dictates otherwise the Weekly_Schedule and Exception_Schedule properties of the Schedule object and the Date_List property of the Calendar object should be modifiable, preferably using standard BACnet services. This allows operator workstations from multiple manufacturers to modify schedules.

10.2 Schedules should be capable of numerous weekday entries and exception entries

The Scheduling BIBBs (K.3.2 in the BACnet standard) require only six entries per day in the Weekly_Schedule and one entry in the Exception_Schedule. For improved usefulness, unless the application dictates otherwise, devices should support more entries than that, especially in the Exception_Schedule.

10.3 Schedule objects should not be used to schedule complex or proprietary datatypes

Schedule objects should not be configured to schedule complex ("constructed") or proprietary datatypes, for better interoperability with operator workstations.

10.4 Schedule objects should schedule certain basic datatypes

The BTL has determined that Schedule objects whose List_Of_Object_Property_References property can be changed, especially if by standard BACnet services, should be able to schedule at least a minimum set of datatypes, for improved interoperability. These datatypes are:

NULL	Unsigned	REAL
BOOLEAN	INTEGER	Enumerated

If a Schedule object supports only SCHED-I-B (that is, it only writes to properties within the same device as the Schedule object), if its List_Of_Object_Property_Reference can be changed and there are any commandable objects in the device, then the Schedule object should be capable of scheduling NULL values.

10.5 Workstations shall be able to configure schedules with NULL datatype

Workstations shall be able to configure Schedule object schedules to write the NULL datatype. This allows the Schedule object to relinquish a value previously written to a priority array.

10.6 Schedule objects should at a minimum be able to schedule BACnetBinaryPV

Unless fixed by the application, Schedule objects should at a minimum be able to schedule the BACnetBinaryPV enumeration, for interoperability.

10.7 Workstations should, at a minimum, support Schedules that schedule REAL, Enumerated, BOOLEAN, and Unsigned32 datatypes

Operator workstations should, at a minimum, be able to view and modify Schedule objects that schedule REAL, Enumerated, BOOLEAN, and Unsigned32 datatypes, and interoperate with Schedules which may contain NULL values.

10.8 Schedule objects scheduling Unsigned and INTEGER values should support 32 bit values

Schedule objects that are able to schedule Unsigned and INTEGER values should be able to schedule, at a minimum, 32 bit values for purposes of interoperability.

10.9 Schedule objects scheduling Enumerated values should support 16 bit values

Schedule objects that are able to schedule Enumerated values should be able to schedule, at a minimum, 16 bit values for purposes of interoperability.

10.10 Support for Protocol Revision 4 Schedule objects requires at least Protocol Revision 4

The BTL requires that a device that supports (that is, can contain) Schedule objects conforming to Protocol_Revision 4 (implemented with the changes appearing in Addendum 135-2001*a*-1, including the Schedule_Default property) shall claim protocol revision 4 or higher in its Device object's property Protocol_Revision. This is required as an unambiguous declaration of the Schedule object's expected operation.

The BTL requires workstations to support (that is, to access Schedule objects in other devices) both Schedule objects conforming to Protocol_Revision 4 as well as objects conforming to earlier versions of the standard. [Note: the old version is published in electronic format only in the ASHRAE Bookstore's superseded standards section]

10.11 NULL is a valid value for the Schedule_Default property

Scheduling prior to revision 4 allowed the use of the NULL value in the TimeValue pair of the Weekly or Exception Schedule in order to indicate that the schedule has completed and control should return to the object. This allows the schedule object to relinquish control of the controlled objects.

In a Protocol_Revision 4 or later Schedule, the only way for the schedule to relinquish control is to place a NULL in the Schedule_Default property.

10.12 Be prepared to read the Exception_Schedule array element by element

Some devices , even those that support segmentation, have Exception_Schedule property values that are too large to transmit in a single APDU, even if segmented. If a device needs to read another's Exception_Schedule property value, be prepared to read it array element by array element.

11 Trending

The following guidelines apply to devices that contain Trend Log objects or that obtain trending data using the ReadRange service.

11.1 Implement the most recent Trend Log and ReadRange

The Trend Log object and ReadRange service shall be implemented as revised in Addendum 135-2001*b* (and as found in BACnet-2004). These are more robust in guaranteeing properly ordered data.

11.2 Support for Protocol Revision 3 Trend Log objects requires at least Protocol Revision 3

The BTL requires that a device that supports Trend Log objects conforming to Protocol_Revision 3 (implemented with the changes appearing in Addendum 135-2001*b*-1) shall claim protocol revision 3 or higher in its Device object's property Protocol_Revision. This is required as an unambiguous declaration of the Trend Log object's expected operation.

11.3 Support for Protocol Revision 7 Trend Log objects requires at least Protocol Revision 7

The BTL requires that a device that supports any of the Trend Log properties Logging_Type, Align_Intervals, Interval_Offset, Trigger, Status_Flags, or Reliability added at Protocol_Revision 7 (in Addendum 135-2004*b*-4) shall claim protocol revision 7 or higher in its Device object's property Protocol_Revision.

11.4 ReadRange clients shall use a 'Count' parameter in the range of INTEGER16

The BACnet standard restricts the range of values used for the 'Count' parameter of the ReadRange service, so that it is unnecessary for the server devices to handle the full BACnet-encodable range of the INTEGER datatype. Client devices shall restrict the range they use to INTEGER16 to ensure interoperation. This restriction should be observed in all BACnet implementations, and is mandated in Addendum 135-2010*ad* in Protocol_Revision 13 of the BACnet standard..

12 Routing

The following guidelines apply to BACnet routers.

12.1 Drop messages with a Hop Count of zero

The BTL requires that if a message with a Hop Count with a value of zero is received, it shall be discarded. (A message with a Hop Count of zero should never be seen on a LAN; this is a defence against a non-conformant router.)

12.2 Drop messages if Hop Count is decremented past zero

To accommodate the intent of clause 6.3.2 of the BACnet standard, the BTL requires that if the Hop Count of a received message has a value less than or equal to the amount by which it is about to be decremented (which means that after being decremented it would have a negative or zero value), the message shall be discarded.

12.3 Initialize-Routing-Table might not operate as expected

The Initialize-Routing-Table message can convey device-specific (proprietary) information. Therefore this message might not operate as expected when conveyed between equipment from different manufacturers.

12.4 Add known networks to empty Router-Busy-To-Network messages

The BACnet standard permits a router to initiate Router-Busy-To-Network messages with an empty list of networks. A router that receives such a message (with an empty list) shall add a list of all networks it knows to be reachable through the initiating router before forwarding the Router-Busy-To-Network to its other ports.

12.5 Routers should redirect unicast messages

The BACnet standard does not discuss the issue of redirecting unicast messages incorrectly sent to the wrong router on a network (that is, the message is sent to a router that is not the router to the destination network). The BTL-WG has determined that if a router receives a message from a device on a network that should have gone to a different device on that network, the router should decrement the Hop Count in the message (if present), add SNET information to the NPCI (if not present), and forward the message to the correct router.

12.6 Routers should not redirect broadcast messages

The BACnet standard allows a device to locate the router to a remote network by broadcasting a confirmed request on the local network and observing the SA conveyed in the response (Clause 6.5.3, paragraph 2). In order to avoid unnecessary duplicate messages, a router should drop confirmed-request messages sent with a broadcast MAC address where the destination network is not reachable through the router.

12.7 Routers should forward unknown network message types

The BACnet standard provides for a Reject-Message-To-Network with a reject reason octet of X’03’, specifying “It is an unknown network layer message type.” This response should only be returned by the device to which the message is addressed, not by an intervening router. (Routers implemented to Protocol_Revision 4 or higher shall convey unknown network layer message types not addressed to them.) This allows new message types to be conveyed across networks using older routers.

12.8 Routers should support the maximum frame size for each supported medium

Routers should always support the maximum frame size for each supported LAN type, as defined in **Table 6-1** of the BACnet standard.

13 Backup and Restore

The following guidelines apply to devices that can have their configurations backed up, or are capable of backing up and restoring other devices' configurations, per clause 19.1 of the BACnet standard.

13.1 Size the configuration File objects before restoring

If the File_Size property value does not match the size of the configuration file content to be restored, clear the contents of the file object by writing 0 to the File_Size property. Then start writing the data in order.

If the client writes a File_Size to the server when the size does not need to be changed, the server may respond with an error. The client shall continue in this case by writing the data to the server in order.

13.2 A device should be able to handle an interrupted Restore operation

In the case where a Restore operation fails, Clause 19.1.3.4 of BACnet-2004 states: "Every attempt shall be made to leave device B in a state that will accept additional restore procedures."

13.3 Empty files shall be backed up and restored if listed in the Configuration_Files property

If a file is listed in the Configuration_Files property it shall be backed up and restored even if its File_Size property indicates a size of zero.

14 Annex J BACnet/IP

The following guidelines apply to devices that implement Annex J of BACnet-2004, BACnet/IP.

14.1 A non-BBMD BACnet/IP device shall be able to register as a Foreign Device

The BTL requires that a BACnet/IP device that does not contain a BBMD shall be able to register as a Foreign Device with a BBMD. This prevents situations where a BACnet/IP device cannot communicate due to an inability to receive broadcasts.

This capability should be configurable to be enabled or disabled by some means, as the device could reside on the same subnet as a BBMD.

14.2 A Broadcast Distribution Table should be able to contain more than four entries

The BTL requires that a BBMD's Broadcast Distribution Table be able to hold at least four entries. If at all possible the BDT should be much larger, able to contain at least 32 entries. This allows the BBMD to fit better in larger installations.

14.3 Two-hop BBMD entries are required – one-hop support is not required

The BTL is only requiring that a BBMD's Broadcast Distribution Table be able to support a netmask of 255.255.255.255. This is in accordance with observed practices and policies for Annex J installations, where one-hop support (from IP routers) is generally disallowed.

14.4 BBMDs shall support Foreign Device registrations

The BTL requires BBMDs to support Foreign Device registrations which imply a connection across a larger internet. This is to ensure that any time a BBMD is present in an Annex J B/IP network foreign devices can connect to the B/IP network.

14.5 A Foreign Device Table should be able to contain at least two entries

The BTL requires that a BBMD's Foreign Device Table be able to hold at least two entries. If at all possible the BTL recommends support for more entries in an FDT.

14.6 A BBMD shall be capable of forwarding Forwarded-NPDU broadcasts to its local subnet

Annex J.4.5, paragraph four, allows a BBMD that receives a Forwarded-NPDU to omit the broadcast on its local subnet (using the B/IP broadcast address) if no other BACnet devices are present on its local subnet. The BTL requires that a BBMD be capable of forwarding the broadcasts in order to support the presence of other BACnet devices on the subnet.

14.7 BACnet/IP devices should support classless addressing

BACnet/IP devices should support classless IP addressing, because this is frequently encountered in installations.

14.8 Routers supporting BACnet/IP shall support fragmentation

Routers which support Annex J BACnet/IP routers shall support fragmentation (and re-assembly) of IP frames. This permits full-sized Ethernet frames (APDU size 1470 octets) to be conveyed across IP networks.

15 Miscellaneous

This section lists items that don't fit anywhere else.

15.1 Context tags greater than 14 shall be properly handled by a parser

Although early versions of the BACnet standard contained no standard context tags greater than 13, it is not unreasonable to expect context tag numbers across the complete range, including extended tags in the range 15 and higher. Parsers shall handle all tag numbers, including those 15 and higher without crashing. The method for encoding tag numbers of 15 and higher is described in clause 20.2.1.2 of the BACnet standard.

15.2 Implement the Abort reasons of Clause 5.4.5.3 (BACnet-2004)

The BTL suggests that the Abort reasons of Clause 5.4.5.3 (in BACnet-2004, introduced in Addendum 135-2001*c*-7) be implemented, regardless of the Protocol_Revision claimed by the device.

15.3 Use of error codes from higher Protocol_Revisions levels may be supported

The BTL suggests that error codes and usages introduced in a later Protocol_Revision may be implemented regardless of the Protocol_Revision claimed by the device.

15.4 BACnet-EIB/KNX mapping may be done independent of Protocol_Revision

The BTL allows a device to implement the BACnet to EIB/KNX mapping defined in Annex H.5 and introduced in Addendum 135-2001*d*-1 without regard to the Protocol_Revision claimed by the device.

15.5 Non-run-time data should survive a power reset

Property values in objects that are not a reflection of runtime status should survive a power reset. For example, relinquish-default, object names, descriptions, etc.

16 Known Issues

This section lists known issues of interpreting the standard.

16.1 Aligned_Intervals of TimeSynchronization, UTCTimeSynchronization, and Trending is aligned to the Local_Time and Local_Date properties

The Align_Intervals property indicates alignment relative to Local_Time, rather than to UTC time, even for the issuance of UTCTimeSynchronization. So both TimeSynchronization and UTCTimeSynchronization would be issued at approximately the same time.

17 Resources

Resources for implementers of BACnet devices are listed here.

17.1 BACnet standards:

ANSI/ASHRAE 135-2012 (the BACnet standard) and **ASHRAE 135.1-2013** (the testing standard) are available in print and downloadable PDF from: http://www.techstreet.com/ashraegate.html All past versions of the BACnet standard and the testing standard are available for purchase from the ASHRAE bookstore. Access to those earlier versions is recommended, to understand the history of changes in the standard, and to read the original wording of clauses which have been revised or deprecated.

Addenda (additions) and **errata** (corrections) to the BACnet standards (ANSI/ASHRAE 135-1995, 2001, 2004, 2008, 2010, and 2012) can be downloaded from http://www.ashrae.org: select "Technology & Standards," then select "Standards Addenda/Errata/Interpretations." Similarly, **Addenda** (additions) and **errata** (corrections) to the BACnet testing of conformance standards (ANSI/ASHRAE 135.1-2003, 2007, 2009, 2011, and 2013) are available there. Reading all past addenda provides a clearer understanding of the history of changes in the standard, and the reasoning for certain of the recommendations specified here to enhance backward compatibility.

Interpretation Requests (clarification of particular issues in the standard) may be downloaded from ASHRAE's website at:

https://www.ashrae.org/standards-research--technology/standards-interpretations/interpretations-for-standard-135-2001 (BACnet 135-2001)

https://www.ashrae.org/standards-research--technology/standards-interpretations/interpretations-for-standard-135-2004 (BACnet 135-2004)

https://www.ashrae.org/standards-research--technology/standards-interpretations/interpretations-for-standard-135-2008 (BACnet 135-2008)

https://www.ashrae.org/standards-research--technology/standards-interpretations/interpretations-for-standard-135-2010 (BACnet 135-2010)

https://www.ashrae.org/standards-research--technology/standards-interpretations/interpretations-for-standard-135-2012 (BACnet 135-2012)

ISO Standard 16484-5:

http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=55404

17.2 The BACnet website

The BACnet committee (ASHRAE/SSPC 135) maintains a website at http://www.bacnet.org. This site presents the latest news from the BACnet world, links to various BACnet organizations, a bibliography of BACnet articles, a list of BACnet Vendor IDs plus vendor contact information, and much more.

17.3 The BACnet Testing Labs website

The BACnet Testing Labs (BTL) has posted a number of Internet links of use to BACnet implementers on its website at: http://bacnetinternational.org/btl. This site also contains information about the BTL's conformance testing and listing program for BACnet devices, as well as the application package for submitting devices for testing.

This site also includes a link for downloading the latest published version of this "BTL Implementation Guidelines".

17.4 Public e-mail lists:

There are several organizational (open to members) and regionally-oriented BACnet e-mail lists maintained by various organizations. There are also a couple of general lists, as follows

BACnet-L	http://www.bacnet.org/Contact/BACnet-L.htm

BACnet-L is a list for general discussions of all things BACnet.

BTL Announcements	http://groups.yahoo.com/group/BTL-announcements/join

The BTL maintains a mailing list for announcing upcoming BTL-related events, such as its Interoperability Testing Workshops (a.k.a. "PlugFests"), the release of new versions of open-source BACnet testing tool: Visual Test Shell (VTS), and upcoming meetings of the BTL working group.