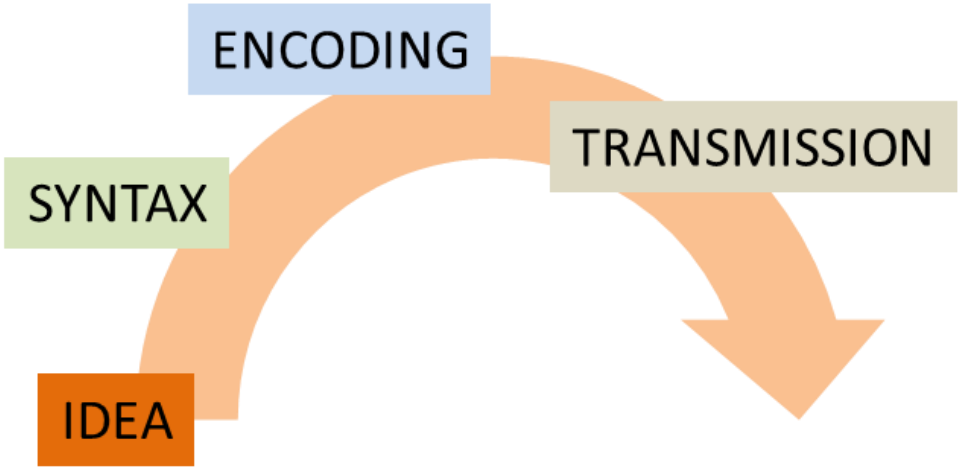
Understanding BACnet MS/TP Encoding

Steve Karg

18-Jan-2012

TUTORIAL



Contents

Understanding MS/TP Encoding

18-Jan-2012

Steve Karg

Introduction

This document shows examples of BACnet messages as they might appear on a BACnet Master-Slave Token Passing (MS/TP) datalink. The examples describe the derivation of the encoded values from the service definitions and encoding rules contained in ANSI/ASHRAE 135-2012, the BACnet standard (referred to herein as "BACnet").

References to numbered "clauses" are to the numbered sections of the BACnet standard ANSI/ASHRAE 135-2012. Although no page numbers are referenced, it is easy to search the electronic version of the standard for the name of an entity or datatype. Note that names change their case and spacing format in the standard, so searching for the Present_Value property in Clause 12 of the BACnet standard is equivalent to searching for the present-value property enumeration in Clause 21.

To construct a BACnet message, perform the following steps:

1. Find the appropriate service description in Clauses 13 to 17. This will describe in English the parameters to be used.
2. Find the ASN.1 definition of the service and its parameters in Clause 21. This is the formal definition of the service request.
3. Use clause 20.1 to encode the APDU header.
4. Use clause 20.2 to encode the tagged portion of the APDU.
5. Use clause 6 to define and encode the network layer header.
6. Use clause 7 to 11, or Annex J, to define and encode the datalink and physical layer information. In the case of MS/TP, we will use clause 9.

The examples which follow show this process.

A Quick Course in ASN.1

This section is intended to assist readers unfamiliar with the ASN.1 notation in reading BACnet clause 21. If you are deeply interested in ASN.1, refer to ISO/IEC 8824 which formally defines ASN.1, and to BACnet clause 20.2 which defines BACnet's encoding of ASN.1. The ASN.1 descriptions in this section are given in tutorial language, rather than in specification language.

Here are some symbols and the names we will use to refer to them:

{ } - braces, **[ ]** - brackets, **( )** - parentheses, **--** double dashes

-- comment text

Double dashes begin comment text that is not part of the definition. The comment is ended by the end of line, similar to C++ "//" comments.

name ::= expression

Defines "name" in terms of the expression.

SEQUENCE { *item item … item* **}**

Means that one of each item between the braces is included, unless the item is marked with the keyword OPTIONAL, in which case it may or may not be included.

SEQUENCE OF *item*

Means a list of zero or more items. The item may be another ASN.1 definition, or it may be a SEQUENCE { }. Note the difference between the one-shot SEQUENCE and the list defined by SEQUENCE OF.

CHOICE { *item item … item* **}**

Means a selection of only one of the items listed between the braces is included.

ENUMERATED {*name (value), ...***}**

defines an enumeration, where a list of "names" between the braces are assigned values shown in parentheses.

[*n***]**

A number in brackets denotes a context tag, which will generally appear in the encoding (the exception being tags defined in the APDU header productions).

ABSTRACT-SYNTAX.&Type

Indicates that a CHOICE of any tagged datatype is valid.

General Notes

BACnet MS/TP, as described in BACnet clause 9, is a peer to peer or slave Datalink protocol. **The MS/TP protocol is described as three Finite State Machines, and must be implemented per the state diagrams and specifications of BACnet (it is only described summarily here)**. The Receive Finite State Machine must be implemented in all cases. The Master Node and Slave Node Finite State Machines both may be implemented in the same device taking care to segregate the station addresses, or one implemented and not the other, or some kind of configuration parameter that indicates what type of node is used. The Master Node is the peer to peer node that passes a token to other Master Nodes to regulate contention on the wire, and can use station addresses 0-127. The Slave Node only emits frames when they are requested, and can use station addresses 0-254. The station address 255 is reserved for broadcasts. Station addresses are normally configured when the device is in service by the installation technician so that each MS/TP segment includes only unique station addresses.

A Master Node must to be able to find a peer node, pass a token to another node, check for new peer nodes, recover after a dropped token, and receive and send data. The Slave node must only be able to receive and send data. These actions must be done in a timely manner or the datalink will not function correctly. Data Frames expecting a reply must be acknowledged within the allotted time (250ms) even if the frame was not expected by the upper stack layers. Frames received that contain a Protocol Data Unit (PDU) not expected by the upper stack layers should be dealt with as described in BACnet clause 5. Appropriate actions will generally be either to discard the PDU, or to respond to it with a BACnet-Abort-PDU.

Attempts to implement an MS/TP Master node will not be successful unless the above concerns are addressed. In particular, implementations must be prepared to receive various broadcast messages at any time.

MS/TP Frame

All MS/TP frames are of the following format:

Preamble 1 (0x55)
Preamble 2 (0xFF)
Frame Type (one byte)
Destination station address (one byte)
Source station address (one byte)
Data Length (2 bytes, most significant byte first)
Header CRC (one byte)
Data (present only if Length is non-zero, from one up to 480 bytes)
Data CRC (present only if Length is non-zero, two bytes, least significant octet first)
(pad) (optional, at most one byte of padding: 0xFF)

The Frame Types are currently defined for 0-7, with 8-127 reserved for ASHRAE. Frame Type 128-255 are available to vendors for non-BACnet frames, with the first two bytes of the Data indicating the vendor identifier. The current defined frames are Token (0), Poll For Master (1), Reply To Poll For Master (2), Test_Request (3), Test_Response (4), BACnet Data Expecting Reply (5), BACnet Data Not Expecting Reply (6), and Reply Postponed (7).

The Destination station address can be 0-127 to send to Master Nodes, 0-254 to send to Slave Nodes, and 255 for Broadcast to all nodes.

The Data Length bytes specifies the number of bytes of Data in the message, and will be zero if there is no Data in this frame.

The Header CRC is calculated as defined in Annex G. In operation, the sender sets the initial content of the Header CRC register to 0xFF. The register is then modified by division by the generator polynomial G(x) of the Frame Type, Destination Station Address, Source Station Address, and Data Length fields. The ones-complement of the resulting remainder is transmitted as the 8-bit Header CRC. The receiver of this packet sets the Header CRC register to 0xFF. The register is then modified by division by the generator polynomial G(x) of the Frame Type, Destination Address, Source Address, Length, and Header CRC fields of the incoming message. A transmission error free message will have a CRC result of 0x55.

The Data CRC is calculated as defined in Annex G. In operation, the sender sets the initial 16-bit Data CRC register to 0xFFFF. The register is then modified by division by the generator polynomial G(x) of the Data field. The ones complement of the resulting remainder is transmitted, least significant octet first, as the 16 bit Data CRC. The receiver sets the initial 16-bit Data CRC register to 0xFFFF. The register is then modified by division by the generator polynomial G(x) of the Data and Data CRC fields of the incoming message. A transmission error free message will have a CRC result of 0xF0B8.

Establishing the MS/TP network

Simplistically speaking, after an MS/TP Master Node initializes, it will wait for 500ms of Silence on the wire before starting to establish an MS/TP network. The node will then generate Poll-For-Master frames to seek peer nodes, and after finding a peer node by receiving a Reply-To-Poll-For-Master frame, it will send the peer a Token frame. After receiving 50 Token frames, the node will begin a maintenance mode that sends a Poll-For-Master seeking peer devices with station addresses between itself and its peer. This cycle will continue forever.

Although the specific MS/TP state machine text defines the exact timing and parameters used, here is a simple description of the messages used when establishing an MS/TP network between Master Node devices. MS/TP networks can be created with up to 128 Master Nodes, but for the sake of brevity, we will describe only two Master Nodes. For this example, the station addresses, which must be field configurable and unique on this MS/TP segment, are 0x01 and 0x03.

MS/TP Poll-For-Master frame from station 1 to station 2

0x55	Preamble
0xFF	Preamble
0x01	Frame Type 1: Poll For Master
0x02	Destination station address
0x01	Source station address
0x00	Data Length
0x00	
0xF5	Header CRC

For this example, station 2 does not exist, so after a brief wait (Tusage_timeout), station 1 will try the next station: station 3.

0x55	Preamble
0xFF	Preamble
0x01	Frame Type 1: Poll For Master
0x03	Destination station address
0x01	Source station address
0x00	Data Length
0x00	
0x7C	Header CRC

Station 3 exists, and replies quickly (within Tusage_delay).

0x55	Preamble
0xFF	Preamble
0x02	Frame Type 2: Reply To Poll For Master
0x01	Destination station address
0x03	Source station address
0x00	Data Length
0x00	
0xD7	Header CRC

Station 1 sees the reply from station 3, and passes the token to station 3.

0x55	Preamble
0xFF	Preamble
0x00	Frame Type 0: Token
0x03	Destination station address
0x01	Source station address
0x00	Data Length
0x00	
0xFA	Header CRC

Station 3 receives the Token, and now must find a peer node at the next highest station address in order to pass the Token in sequence. Station 3 will continue to seek a peer node until it runs out of station addresses. It will limit itself to the highest MAC address (Max_Master) and wrap around to the beginning until reaching its own MAC address.

0x55	Preamble
0xFF	Preamble
0x01	Frame Type 1: Poll For Master
0x04	Destination station address
0x03	Source station address
0x00	Data Length
0x00	
0xF5	Header CRC

...

0x55	Preamble
0xFF	Preamble
0x01	Frame Type 1: Poll For Master
0x00	Destination station address
0x03	Source station address
0x00	Data Length
0x00	
0xD7	Header CRC

Station 3 did not find any other peers, so it passes the Token to Station 1, the node it already knows exists.

0x55	Preamble
0xFF	Preamble
0x00	Frame Type 0: Token
0x01	Destination station address
0x03	Source station address
0x00	Data Length
0x00	
0xD8	Header CRC

Station 1 and Station 3 continue to pass the token between themselves, and after each node sees a number of tokens (Npoll), execute a Poll-For-Master to discover any new peers that may have been added to the network, or that may have come online.

Data Expecting and Not Expecting Reply

The MS/TP protocol includes frame types for sending BACnet frames that include "data". The data, in our case, is going to be constructed from a BACnet NPDU (network header) and optionally a BACnet APDU (application layer data). In general, BACnet confirmed services use a Data Expecting Reply frame type, and BACnet unconfirmed services use a Data Not expecting Reply frame type.

The Network Layer

The network layer, or NPDU, provides a means for messages to be routed from one BACnet network to another, even if the data link layer is different (i.e. routing from BACnet/IP to MS/TP). The Network Layer is specified in clause 6, and the NPDU encoding and decoding is necessary to understand the data portion of the message and the length of the data. Here is an example of the most simplistic NPDU:

	NPDU	
0x01	Network protocol version	
0x00	Network control octet (clause 6.2.2)	
	Bit7 = 0	"BACnet APDU"
	Bit6 = 0	not used
	Bit5 = 0	DNET not present
	Bit4 = 0	not used
	Bit3 = 0	SNET not present
	Bit2 = 0	no reply expected
	Bit1,0= 00	normal priority

The Network control octet indicates which optional NPDU fields are included in the message, which alters the length of the NPDU and determines where the application begins.

The Application Layer

The application layer, or APDU, handles the peer-to-peer interactions with a remote application layer in another BACnet device. These interactions involve information exchange, and are known in BACnet as abstract service primitives, or services. The Application Layer is specified in clause 5, and the APDU encoding and decoding is defined there and in clause 20 and clause 21.

Unconfirmed Services (Who-Is and I-Am)

BACnet defines unconfirmed application services based on a client and server communication model. A client requests service from a server via a particular service request instance. The server provides service to a client and responds to the request.

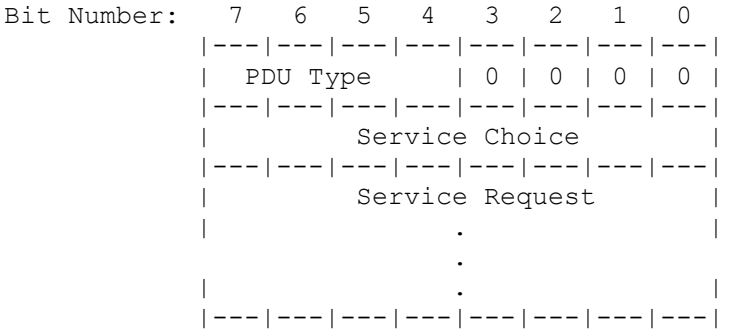
The BACnet-user that assumes the role of a client is called the "sending BACnet-user" and the BACnet-user that assumes the role of the server is called the "receiving BACnet-user".

A "BACnetDevice" is defined as "any device, real or virtual, that supports digital communication using the BACnet protocol. Each BACnet Device contains exactly one Device object, as defined in clause 12. A BACnet Device is uniquely located by an NSAP, which consists of a network number and a MAC address. In most cases, a physical device will implement a single BACnet Device. It is possible, however, that a single physical device may implement a number of "virtual" BACnet Devices."

The Who-Is and I-Am services are used to discover a BACnet device NSAP (address). This is also known as dynamically binding Device IDs (Device Object Identifiers) to station addresses. The Who-Is and I-Am services can also be used for simply discovering the devices on a BACnet network. Since these messages are usually globally broadcast, your device must be prepared to see these messages at any time.

Who-Is

The Who-Is and I-Am services are described in clause 16.10. The ASN.1 definition of these services is in clause 21. Who-Is and I-Am are Unconfirmed services. A BACnet-Unconfirmed-Request-PDU is defined in clause 20.1.3 as a service-choice followed by a service-request. The comment states that tags are not used in the header encoding. The format of the BACnet-Unconfirmed-Request-PDU is shown in clause 20.1.3.3:



The service-choice is defined to be of type BACnetUnconfirmedServiceChoice, which is defined in Clause 21 as an enumeration. The service-choice value for who-Is is 8.

The service-request is defined to be of type BACnetUnconfirmedServiceRequest, which is defined in Clause 21. The CHOICE for who-Is is the production Who-Is-Request, which is defined in Clause 21 as a SEQUENCE consisting of an OPTIONAL pair of unsigned integers which specify the range of devices which should respond.

Who-Is-Request ::= SEQUENCE {

deviceInstanceRangeLowLimit [0] Unsigned (0..4194303) OPTIONAL, -- must be used as a pair, see 16.10

deviceInstanceRangeHighLimit [1] Unsigned (0..4194303) OPTIONAL -- must be used as a pair, see 16.10

}

The encoding of unsigned integers are primitive, and their encoding is defined in clause 20.2.4. In this case, the ASN.1 "[0]" and "[1]" specifies that encoding should contain context tag 0 or 1 respectively, rather than the application tag 2 which would be used if the ASN.1 did not contain the "[0]" or "[1]". Clause 20.2.15 subitem "a)" and "b)" specify the encoding of such a value to be the context tag, followed by the data octets encoded as specified in clause 20.2.4.

In the example which follows, the unsigned integers are absent, meaning (according to clause 16.9.2) that all devices should respond with I-Am. The network header specifies a global broadcast (DNET 0xFFFF, DADR of length 0). The network header contains no source network or address for the requester, as the requester is assumed in this example to reside on the same MS/TP network where the request is observed.

Note that Unconfirmed services on MS/TP use a **BACnet Data Not Expecting Reply** frame, and the node must wait for a **Token** frame before sending this type of frame.

MS/TP REQUESTER sends

	MS/TP header
0x55	Preamble
0xFF	Preamble
0x06	Frame Type: BACnet Data Not Expecting Reply (6)
0xFF	Destination station address (broadcast)
0x01	Source station address
0x00	Data Length = 21
0x15	
0x8E	Header CRC
	NPDU
0x01	Network protocol version
0x20	Network control octet (clause 6.2.2)
	Bit7 = 0 "BACnet APDU"
	Bit6 = 0 not used
	Bit5 = 1 DNET present
	Bit4 = 0 not used
	Bit3 = 0 SNET not present
	Bit2 = 0 no reply expected
	Bit1,0 = 00 normal priority
0xFF	DNET 65535 (broadcast)
0xFF	
0x00	DLEN 0 (denotes broadcast)
0xFF	Hop Count 255 (clause 6.2.3)
	APDU
0x10	APDU type 1: Unconfirmed Request
0x08	Service Choice 8: Who Is
	No Tagged Service parameters for this example
	MS/TP footer
0x15	Data CRC-16
0xB6	

I-Am

When a BACnet device receives a Who-Is whose device instance range limits include the Device Identifier, or a Who-Is which does not contain device instance range limits, the device will respond by with an I-Am.

The I-Am service is described in clause 16. The ASN.1 definition is in clause 21. This is an Unconfirmed service. A BACnet-Unconfirmed-Request-PDU, defined in clause 20.1.3, is defined to be a service-choice followed by a service-request. The comment states that tags are not used in the header encoding. Header encoding for BACnet-Unconfirmed-Request-PDU is shown in clause 20.1.3.3.

The service-choice is defined to be of type BACnetUnconfirmedServiceChoice, which is defined in clause 21 as an enumeration. The value for I-Am is 0.

The service-request is defined to be of type BACnetUnconfirmedServiceRequest, which is defined in clause 21. The CHOICE for I-Am is the production I-Am-Request, which is defined on in clause 21, as a SEQUENCE consisting of a BACnetObjectIdentifier containing the device's identifier, an unsigned integer specifying the maximum APDU length receivable by the device, an enumeration specifying the device's segmentation abilities, and an unsigned integer specifying the device's vendor.

I-Am-Request ::= SEQUENCE {

iAmDeviceIdentifier	BACnetObjectIdentifier,
maxAPDULengthAccepted	Unsigned,
segmentationSupported	BACnetSegmentation,
vendorID	Unsigned16

}

The encoding of a BACnetObjectIdentifier is defined in clause 20.2.14. In this example, the Device Object has Object Identifier objectType = 8 (Device), instance = 1. The encoding of an unsigned integer is defined in clause 20.2.4. In this example, the maximum APDU length is 480. The encoding of an enumeration is defined in clause 20.2.11. In this example, the device cannot send or receive segmented messages, so the value is "segmented-none". In this example, the value of the vendor identifier is 555.

Note that the network header for MS/TP specifies a global broadcast destination. As mentioned earlier, Unconfirmed services on MS/TP use a **BACnet Data Not Expecting Reply** frame, and the node must wait for a **Token** frame before sending this type of frame.

MS/TP RECEIVER sends

	MS/TP header
0x55	Preamble
0xFF	Preamble
0x06	Frame Type: BACnet Data Not Expecting Reply (6)
0xFF	Destination station address (broadcast)
0x03	Source station address
0x00	Data Length = 22
0x16	
0x8E	Header CRC
	NPDU
0x01	Network version 1

0x20	Network control octet
	Bit7 = 0 "BACnet APDU"
	Bit6 = 0 not used
	Bit5 = 1 DNET present
	Bit4 = 0 not used
	Bit3 = 0 SNET not present
	Bit2 = 0 no reply expected
	Bit1,0 = 00 normal priority
0xFF	DNET 0xFFFF (broadcast)
0xFF	
0x00	DLEN 0 (denotes broadcast)
0xFF	Hop Count 255
	APDU
0x10	APDU type Unconfirmed Request
0x00	Service Choice 0: I Am
	Tagged Service parameters follow
0xC4	Application Tag 12 (Object Identifier), length 4
0x02	Object type 8 (Device), Object Instance 1
0x00	
0x00	
0x01	
0x22	Application Tag 2 (Unsigned Integer), length 2
0x01	Maximum APDU size: 480
0xE0	
0x91	Application Tag 9 (Enumerated), length 1
0x03	segmentation: no-segmentation
0x22	Application Tag 2 (Unsigned Integer), length 2
0x02	value 555 = vendorID
0x2B	
	MS/TP footer
0x02	Data CRC-16
0xA8	

Confirmed Services (ReadProperty and WriteProperty)

BACnet defines confirmed application services based on a client and server communication model. A client requests service from a server via a particular service request instance. The server provides service to a client and responds to the request.

The BACnet-user that assumes the role of a client is called the "requesting BACnet-user" and the BACnet-user that assumes the role of the server is called the "responding BACnet-user."

Here is an example of ReadProperty for the present-value property of Analog Input Object number 1.

ReadProperty-Request

The Analog Input Object and its properties are described in clause 12. The ReadProperty service is described in clause 15. The ASN.1 definition of ReadProperty is in clause 21.

A ReadProperty-Request is a Confirmed service as defined in clause 21 BACnet-Confirmed-Service-Request list of choices. A BACnet-Confirmed-Service-Request is part of the SEQUENCE in a BACnet-Confirmed-Request-PDU defined in clause 20.1.2:

```
BACnet-Confirmed-Request-PDU ::= SEQUENCE {
    pdu-type                        [0] Unsigned (0..15), -- 0 for this PDU type
    segmented-message               [1] BOOLEAN,
    more-follows                    [2] BOOLEAN,
    segmented-response-accepted     [3] BOOLEAN,
    reserved                        [4] Unsigned (0..3), -- shall be set to zero
    max-segments-accepted           [5] Unsigned (0..7), -- as per Clause 20.1.2.4
    max-apdu-length-accepted        [6] Unsigned (0..15), -- as per Clause 20.1.2.5
    invoke-id                       [7] Unsigned (0..255),
    sequence-number                 [8] Unsigned (0..255) OPTIONAL, -- only if segmented
    Proposed-window-size             [9] Unsigned (1..127) OPTIONAL, -- only if segmented
    service-choice                  [10] BACnetConfirmedServiceChoice,
    service-request                 [11] BACnet-Confirmed-Service-Request
    -- Context specific tags 0..11 are NOT used in header encoding
}
```

Values may be determined as follows:

segmented-message

BOOLEAN, TRUE if this is a segmented message, FALSE otherwise. FALSE in this example.

more-follows

BOOLEAN, TRUE if this is a segmented message, and this is not the final segment. FALSE otherwise. FALSE in this example.

segmented-message-accepted

BOOLEAN, TRUE if the requester will accept a segmented BACnet-Complex-ACK-PDU as a response, FALSE otherwise. TRUE in this example.

The RECEIVER will accept segmented BACnet-Confirmed-Request-PDUs, and is capable of returning segmented BACnet-ComplexACK-PDUs. Support of segmentation by requesters is optional.

maximum-APDU-size-accepted

Specifies the maximum size of a single PDU or PDU segment. In this example, an enumerated value of 3, denoting a maximum of 480 octets. This is the maximum value allowed for BACnet PDUs on MS/TP, and is the maximum value accepted by the this device.

invokeID

A message "serial number" used by the requester to associate the response with the request. All segments of a segmented request must have the same invokeID. The invokeID should be incremented for each new BACnet-Confirmed-Request-PDU. This facilitates debugging, as it allows responses to be correlated with the requests which caused them. The invokeID will increment in each example which follows.

sequence-number

Identifies the segments of a segmented BACnet-Confirmed-Request-PDU. Not present in un-segmented requests, such as this example.

proposed-window-size

Specifies the maximum segmentation window size acceptable to a requester sending a segmented BACnet-Confirmed-Request-PDU. Not present in un-segmented requests, such as this example.

service-choice

The service-choice is defined to be of type BACnetConfirmedServiceChoice, which is defined in clause 21 as an enumeration. The value for readProperty is 12 decimal.

service-request

The service-request is defined to be of type BACnet-Confirmed-Service-Request, which is defined in clause 21. The CHOICE for readProperty is the production ReadProperty-Request, which is defined in clause 21 as a SEQUENCE of items:

ReadProperty-Request ::= SEQUENCE {
objectIdentifier [0] BACnetObjectIdentifier,
propertyIdentifier [1] BACnetPropertyIdentifier,
propertyArrayIndex [2] Unsigned OPTIONAL --used only with array datatype
-- if omitted with an array the entire array is referenced
}

The items are "objectIdentifier", tagged with context tag 0 and of type BACnetObjectIdentifier; "propertyIdentifier", tagged with context tag 1 and of type BACnetPropertyIdentifier; and "propertyArrayIndex", an OPTIONAL unsigned integer tagged with context tag 2. Since the present-value of an Analog Input Object is not an array, the "propertyArrayIndex" will not be present in this example.

Although BACnet-Confirmed-Request-PDU is defined in Clause 20 to be a SEQUENCE of items, the actual format of the BACnet-Confirmed-Request-PDU is defined in clause 20.1.2.11:

```
Bit Number:   7   6   5   4   3   2   1   0
            |---|---|---|---|---|---|---|---|
            |   PDU Type    |SEG|MOR| SA| 0 |
            |---|---|---|---|---|---|---|---|
            | 0 |  Max Segs |   Max Resp    |
            |---|---|---|---|---|---|---|---|
            |           Invoke ID           |
            |---|---|---|---|---|---|---|---|
            |         Sequence Number       | Only present if SEG = 1
            |---|---|---|---|---|---|---|---|
            |       Proposed Window Size    | Only present if SEG = 1
            |---|---|---|---|---|---|---|---|
            |        Service Choice         |
            |---|---|---|---|---|---|---|---|
            |        Service Request        |
            |               .               |
                            .
            |               .               |
            |---|---|---|---|---|---|---|---|
```

For a ReadProperty-Request, the BACnetPropertyIdentifier is an enumeration defined in clause 21. The present-value enumeration is defined as 85. Enumerations are primitive, and their encoding is described in clause 20.2.11. In this case, the ASN.1 "[0]" specifies that encoding

should contain context tag 0, rather than the application tag 12 which would be used if the ASN.1 did not contain the "[0]". Clause 20.2.15 subitem "a)" and "b)" specify the encoding of such a value to be the context tag, followed by the data octets encoded as specified in clause 20.1.11.

MS/TP REQUESTER sends

	MS/TP header
0x55	Preamble
0xFF	Preamble
0x05	Frame Type: BACnet Data Expecting Reply (5)
0x03	Destination station address
0x01	Source station address
0x00	Data Length = 13
0x0D	
0x98	Header CRC
	NPDU
0x01	Network version 1
0x04	Network control octet
	Bit7 = 0 "BACnet APDU"
	Bit6 = 0 not used
	Bit5 = 0 DNET not present
	Bit4 = 0 not used
	Bit3 = 0 SNET not present
	Bit2 = 1 reply expected (confirmed service request)
	Bit1,0= 00 normal priority
	APDU
0x02	APDU type 0: Confirmed Request; Bit1: Segmented Response Accepted
0x03	Max response 3: up to 480 octets
0x00	Invoke ID 0
0x0C	Service Choice 12: Read Property
	Tagged Service parameters follow
0x0C	Context Tag 0, length 4: Object Identifier
0x00	Data: Object type 0 (Analog Input), Object Instance 1
0x00	
0x00	
0x01	
0x19	Context Tag 1, length 1: BACnetPropertyIdentifier
0x55	Data: Property 85: present-value
	MS/TP footer
0x02	Data CRC-16
0xA8	

ReadProperty - Response

When a BACnet device receives a ReadProperty-Request, it services the request as described in clause 15, and specifically, the Service Procedure in clause 15.5.2. The normal response is a Result(+), which is defined in Table 15-5. Abnormal responses are Result (-), which can occur from malformed service parameters or properties that are unknown or not valid or not readable. If the service parameters are malformed, then a Reject PDU or Abort PDU will be returned. If one of the errors listed under Error Type in clause 15.5.1.3.1 occurs, then an Error PDU will be returned.

Since the response includes data, it must be returned as a BACnet-ComplexACK-PDU, defined in 20.1.5:

```
BACnet-ComplexACK-PDU ::= SEQUENCE {
    pdu-type                [0] Unsigned (0..15), -- 3 for this PDU type
    segmented-message       [1] BOOLEAN,
    more-follows            [2] BOOLEAN,
    reserved                [3] Unsigned (0..3), -- shall be set to zero
    original-invoke-id      [4] Unsigned (0..255),
    sequence-number         [5] Unsigned (0..255) OPTIONAL, --only if segment
    proposed-window-size    [6] Unsigned (1..127) OPTIONAL, -- only if segment
    service-ack-choice      [7] BACnetConfirmedServiceChoice,
    service-ack             [8] BACnet-Confirmed-Service-ACK
    -- Context specific tags 0..8 are NOT used in header encoding
}
```

Values may be determined as follows:

segmented-message

BOOLEAN, TRUE if this is a segmented response, FALSE otherwise. FALSE in this example.

more-follows

BOOLEAN, TRUE if this is a segmented response, and this is not the final segment. FALSE otherwise. FALSE in this example.

original-invokeID

Will contain the invokeID of the BACnet-Confirmed-Request-PDU which is being responded to.

sequence-number

Identifies the segments of a segmented BACnet-Confirmed-Request-PDU. Not present in un-segmented requests, such as this example.

proposed-window-size

Specifies the maximum segmentation window size acceptable to a responder sending a segmented BACnet-ComplexACK-PDU. Not present in un-segmented ComplexACKs, such as this example.

service-ACK-choice

The service-ACK-choice is defined to be of type BACnetConfirmedServiceChoice, which is defined on page 350 as an enumeration. The value for readProperty is 12 decimal.

service-ACK

The service-ACK is defined to be of type BACnet-Confirmed-Service-ACK, which is defined in clause 21. The CHOICE for readProperty is the production ReadProperty-ACK.

```
ReadProperty-ACK ::= SEQUENCE {
    objectIdentifier        [0] BACnetObjectIdentifier,
    propertyIdentifier      [1] BACnetPropertyIdentifier,
    propertyArrayIndex      [2] Unsigned OPTIONAL, --used only with array datatype
                                -- if omitted with an array the entire array is referenced
    propertyValue           [3] ABSTRACT-SYNTAX.&Type
        }
```

ReadProperty-ACK is defined in clause 21 as a SEQUENCE consisting of "objectIdentifier", tagged with context tag 0 and of type Object Identifier; "propertyIdentifier", tagged with context tag 1 and of type BACnetPropertyIdentifier; "propertyArrayIndex", an OPTIONAL unsigned integer tagged with context tag 2; and "propertyValue" tagged with context tag 3 and of type ABSTRACT-SYNTAX.&Type.

Although BACnet-ComplexACK-PDU is defined in Clause 20 to be a SEQUENCE of items, the actual format of the BACnet-ComplexACK-PDU is defined in 20.1.5.8:

```
Bit Number:   7   6   5   4   3   2   1   0
            |---|---|---|---|---|---|---|---|
            |  PDU Type     |SEG|MOR| 0 | 0 |
            |---|---|---|---|---|---|---|---|
            |       Original Invoke ID      |
            |---|---|---|---|---|---|---|---|
            |         Sequence Number       | Only present if SEG = 1
            |---|---|---|---|---|---|---|---|
            |       Proposed Window Size    | Only present if SEG = 1
            |---|---|---|---|---|---|---|---|
            |        Service ACK Choice     |
            |---|---|---|---|---|---|---|---|
            |           Service ACK         |
            |               .               |
                            .
            |               .               |
            |---|---|---|---|---|---|---|---|
```

In this example, a propertyValue will be returned. The designation "ABSTRACT-SYNTAX.&Type" means that the encoding (defined in clause 20.2.19) is the complete encoding including tags of the data type defined for the property value being returned. Since such an encoding includes tags, the ABSTRACT-SYNTAX.&Type is "constructed" according to the definition of 20.2.1.3.2, the context tag 3 must be encoded as an opening tag/closing tag pair surrounding the encoding of the ABSTRACT-SYNTAX.&Type. In this example, the value is a real number.

MS/TP Response to Confirmed Service

The MS/TP frame BACnet Data Expecting Reply (5) allows an MS/TP node up to 250ms to respond with a BACnet Data Not Expecting Reply (6) frame. If the MS/TP device is not able to respond within that time limit with the reply, it must reply with a Reply Postponed (7) frame, and wait for its next Token (0) frame to send the reply.

MS/TP RECEIVER sends Reply Postponed

	MS/TP header
0x55	Preamble
0xFF	Preamble
0x07	Frame Type: Reply Postpone (7)
0x01	Destination station address (unicast)
0x03	Source station address
0x00	Data Length = 0
0x00	
0x4F	Header CRC

MS/TP RECEIVER sends ComplexACK

	MS/TP header
0x55	Preamble
0xFF	Preamble
0x06	Frame Type: BACnet Data Not Expecting Reply (6)
0x01	Destination station address (unicast)
0x03	Source station address
0x00	Data Length = 19
0x13	
0x39	Header CRC
	NPDU
0x01	Network version 1
0x00	Network control octet
	Bit7 = 0 "BACnet APDU"
	Bit6 = 0 not used
	Bit5 = 0 DNET not present
	Bit4 = 0 not used
	Bit3 = 0 SNET not present
	Bit2 = 0 no reply expected
	Bit1,0 = 00 normal priority
	APDU
0x30	APDU type 3: Complex ACK
0x00	Invoke ID 0
0x0C	Service Choice 12: Read Property ACK
	Tagged Service parameters follow
0x0C	Context Tag 0, length 4: Object Identifier
0x00	Data: Object type 0 (Analog Input), Object Instance 1
0x00	
0x00	
0x01	
0x19	Context Tag 1, length 1: BACnetPropertyIdentifier
0x55	Data: Property 85: present-value
0x3E	Context Tag 3, opening tag: ABSTRACT-SYNTAX.&Type
0x44	Application Tag 4 (Real), length 4
0x42	Data: value = 46.4
0x39	
0x99	
0x9A	
0x3F	Context Tag 3, closing tag
	MS/TP footer
0x36	Data CRC-16
0xC6	

WriteProperty - Single Property of Binary Output

Example of a WriteProperty for the present-value property of Binary Output Object instance number 1.

WriteProperty - Request

The Binary Output Object and its properties are described in clause 12. The WriteProperty service is described in clause 15, and specifically, the Service Procedure in clause 15.9.2. The ASN.1 definition of WriteProperty is in clause 21. This is a Confirmed service. The BACnet-Confirmed-Request-PDU is defined to be a SEQUENCE of items. The comment states that tags are not used in the header encoding. Header encoding for a BACnet-Confirmed-Request-PDU is shown in clause 20.1.2.

The service-choice is defined to be of type BACnetConfirmedServiceChoice, which is defined in Clause 21 as an enumeration. The value for writeProperty is 15 decimal.

The service-request is defined to be of type BACnet-Confirmed-Service-Request, which is defined in clause 21 as a sequence of items:

```
WriteProperty-Request ::= SEQUENCE {
    objectIdentifier        [0] BACnetObjectIdentifier,
    propertyIdentifier      [1] BACnetPropertyIdentifier,
    propertyArrayIndex      [2] Unsigned OPTIONAL,
                                --used only with array datatype
                                -- if omitted with an array the entire
                                -- array is referenced
    propertyValue           [3] ABSTRACT-SYNTAX.&Type,
    priority                [4] Unsigned (1..16) OPTIONAL
                                --used only when property is commandable
        }
```

The CHOICE for writeProperty is the production WriteProperty-Request, which is defined as a SEQUENCE consisting of "objectIdentifier", tagged with context tag 0 and of type BACnetObjectIdentifier; "propertyIdentifier", tagged with context tag 1 and of type BACnetPropertyIdentifier; "propertyArrayIndex", an OPTIONAL unsigned integer tagged with context tag 2; "propertyvalue", tagged with context tag 3 and of type ABSTRACT-SYNTAX.&Type; and "priority", an OPTIONAL unsigned integer in the range 1 to 16 tagged with context tag 4. Since the present-value property of a Binary Output Object is not an array, the OPTIONAL "propertyArrayIndex" will not be present in this example.

Since the present-value of a Binary Output Object is a commandable property, "priority" will be present in this example. If the optional "priority" were missing in the service-request, it is assumed to be 16. If the object specificed in "objectIdentifier" is not commandable, the "priority" is ignored. Any objects that include a Priority_Array property are commandable, along with the Channel object. Commandability is defined in clause 19.

The BACnetPropertyIdentifier is an enumeration defined in clause 21. The present-value property has an enumerated value of 85.

In this example, the value to written will be "inactive" (0), and the priority will be 7 (shown in Table 19-1 in clause 19 as available for general use).

MS/TP REQUESTER sends

	MS/TP header
0x55	Preamble
0xFF	Preamble
0x06	Frame Type: BACnet Data Expecting Reply (5)
0x03	Destination station address (unicast)
0x01	Source station address
0x00	Data Length = 19
0x13	
0x1B	Header CRC
	NPDU
0x01	Network version 1
0x04	Network control octet
	Bit7 = 0 "BACnet APDU"
	Bit6 = 0 not used
	Bit5 = 0 DNET not present
	Bit4 = 0 not used
	Bit3 = 0 SNET not present
	Bit2 = 1 reply expected (confirmed service request)
	Bit1,0= 00 normal priority
	APDU
0x02	APDU type 0: Confirmed Request; Bit1: Segmented Response Accepted
0x03	Max response 3: up to 480 octets
0x05	Invoke ID 5
0x0F	Service Choice 15: Write Property
	Tagged Service parameters follow
0x0C	Context Tag 0, length 4: Object Identifier
0x01	Data: Object type 4 (Binary Output), Object Instance 1
0x00	
0x00	
0x01	
0x19	Context Tag 1, length 1: BACnetPropertyIdentifier
0x55	Data: Property 85: present-value
0x3E	Context Tag 3, opening tag: ABSTRACT-SYNTAX.&Type (value)
0x91	Application tag 9 (enumerated), length 1
0x00	Data: enumerated value 0: inactive
0x3F	Context Tag 3, closing tag
0x49	Context Tag 4, length 1: unsigned (priority)
0x07	Data: value 7
	MS/TP footer
0x74	Data CRC-16
0x30	

WriteProperty - Response

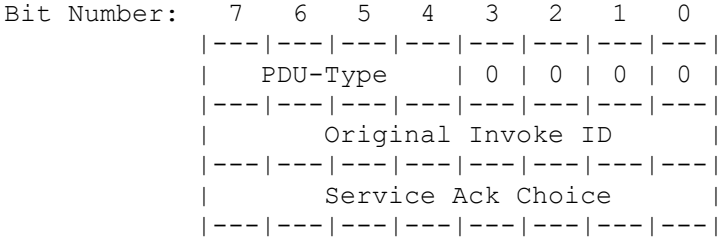
When the RECEIVER receives a WriteProperty-Request, it services the request as described in clause 15.9.2. The normal response is a Result(+), which is defined in Table 15-15 as containing no parameters. The definition in ASN.1 is in clause 21.

Since the response to WriteProperty service does not include data, it must be returned as a SimpleACK-PDU. BACnet-SimpleACK-PDU is defined in clause 21 to be a SEQUENCE of an invokeID and a service-ACK-choice:

BACnet-SimpleACK-PDU ::= SEQUENCE {

	pdu-type	[0] Unsigned (0..15), -- 2 for this PDU type
	reserved	[1] Unsigned (0..15), -- must be set to zero
	invokeID	[2] Unsigned (0..255),
	service-ACK-choice	[3] BACnetConfirmedServiceChoice
		-- Context-specific tags 0..3 are NOT used in header encoding
}		

Note that the comment states that tags are not used in the header encoding. The format of the BACnet-SimpleACK-PDU is shown in clause 20.1.4 and is always three octets long:



The service-ACK-choice is defined to be of type BACnetConfirmedServiceChoice, which is defined in clause 21 as an enumeration. The value for writeProperty is 15 decimal.

MS/TP RECEIVER responds

	MS/TP header
0x55	Preamble
0xFF	Preamble
0x06	Frame Type: BACnet Data Not Expecting Reply (6)
0x01	Destination station address (unicast)
0x03	Source station address
0x00	Data Length = 5
0x05	
0xCA	Header CRC
	NPDU
0x01	Network version 1
0x00	Network control octet
	Bit7 = 0 "BACnet APDU"
	Bit6 = 0 not used
	Bit5 = 0 DNET not present
	Bit4 = 0 not used
	Bit3 = 0 SNET not present
	Bit2 = 0 no reply expected
	Bit1,0= 00 normal priority
	APDU
0x20	APDU type 2: Simple ACK
0x05	Invoke ID 5
0x0F	ServiceACK Choice 15: Write Property
	MS/TP footer
0x47	Data CRC-16
0x41	

Error APDUs

If a REQUESTER sends a correctly encoded Confirmed-Service-Request, and the receiving device encounters some condition while servicing the request which requires the return of Result(-), the RECEIVER will return a BACnet-Error-PDU. For example, when the device receives a WriteProperty-Request, it services the request as described in clause 15.9.2. If a specified object does not exist, or a specified property is not writable, the response is a Result(-), which is defined in Table 15-15 as containing one parameter, Error Type. The definition in ASN.1 is in clause 21.

Since the response is a Result(-), it must be returned as a BACnet-Error-PDU. In clause 20.1.7, BACnet-Error-PDU is defined:

```
BACnet-Error-PDU ::= SEQUENCE {
        pdu-type            [0] Unsigned (0..15), -- 5 for this PDU type
        reserved            [1] Unsigned (0..15), -- shall be set to zero
        original-invoke-id  [2] Unsigned (0..255),
        error-choice        [3] BACnetConfirmedServiceChoice,
        error               [4] BACnet-Error
        -- Context specific tags 0..4 are NOT used in header encoding
        }
```

The error-choice is defined to be of type BACnetConfirmedServiceChoice, which is defined in Clause 21 as an enumeration. The value for write-property is 15 decimal.

The "error" is defined to be of type BACnet-Error, which is defined in Clause 21 as a CHOICE. The value of the CHOICE for write-property, defined to be of type Error. The comment on BACnet-Error states:

-- Context-specific tags 0..31 and 127 are NOT used in the encoding.

-- The tag number is transferred as the error-choice

-- parameter in the BACnet-Error-PDU

The format of the BACnet-Error-PDU is defined in clause 20.1.7.4

```
Bit Number:  7   6   5   4   3   2   1   0
            |---|---|---|---|---|---|---|---|
            |    PDU Type   | 0 | 0 | 0 | 0 |
            |---|---|---|---|---|---|---|---|
            |      Original Invoke ID       |
            |---|---|---|---|---|---|---|---|
            |          Error Choice         |
            |---|---|---|---|---|---|---|---|
            |              Error            |
            |               .               |
                            .
            |               .               |
            |---|---|---|---|---|---|---|---|
```

Error is defined in Clause 21 as a sequence consisting of two enumerations: error-class and error-code. In the case of a non-existent object, the error-class will be "object" and the error-code will be "unknown-object".

In this example, the error-class will be "object" and the error-code will be "unknown-object":

	MS/TP header
0x55	Preamble
0xFF	Preamble
0x06	Frame Type: BACnet Data Not Expecting Reply (6)
0x01	Destination station address (unicast)
0x03	Source station address
0x00	Data Length = 5
0x05	
0xCA	Header CRC
	NPDU
0x01	Network version 1
0x00	Network control octet
	Bit7 = 0 "BACnet APDU"
	Bit6 = 0 not used
	Bit5 = 0 DNET not present
	Bit4 = 0 not used
	Bit3 = 0 SNET not present
	Bit2 = 0 no reply expected
	Bit1,0= 00 normal priority
	APDU
0x50	APDU type 5: Error
0x05	Original Invoke ID 5
0x0F	Error Choice 15 (escaped): Write Property
0x91	Application Tag 9 (enumerated), length 1
0x01	Data: value 1: "object"
0x91	Application Tag 9 (enumerated), length 1
0x1F	Data: value 31: "unknown-object"
	MS/TP footer
0x47	Data CRC-16
0x41	

Reject of Invalid APDUs

If a REQUESTER sends a Confirmed-Service-Request with a valid APDU header, but with errors in the tagged portion (for example, missing items, incorrect tag numbers, or incorrect encoding of data values), the RECEIVER will return a BACnet-Reject-PDU. The BACnet-Reject-PDU is defined in clause 20.1.8, BACnet-Reject-PDU.

```
BACnet-Reject-PDU ::= SEQUENCE {
        pdu-type            [0] Unsigned (0..15), -- 6 for this PDU type
        reserved            [1] Unsigned (0..15), -- shall be set to zero
        original-invoke-id  [2] Unsigned (0..255),
        reject reason       [3] BACnetRejectReason
        -- Context specific tags 0..3 are NOT used in header encoding
        }
```

The format of the BACnet-Reject-PDU is defined in clause 20.1.8.3, and is always three octets long:

```
Bit Number:   7   6   5   4   3   2   1   0
            |---|---|---|---|---|---|---|---|
            |    PDU-Type   | 0 | 0 | 0 | 0 |
            |---|---|---|---|---|---|---|---|
            |       Original Invoke ID      |
            |---|---|---|---|---|---|---|---|
            |         Reject Reason         |
            |---|---|---|---|---|---|---|---|
```

The BACnetRejectReason enumeration is defined in Clause 21.

In this example, the reject reason will be "invalid-tag":

	MS/TP header
0x55	Preamble
0xFF	Preamble
0x06	Frame Type: BACnet Data Not Expecting Reply (6)
0x01	Destination station address (unicast)
0x03	Source station address
0x00	Data Length = 5
0x05	
0xCA	Header CRC
	NPDU
0x01	Network version 1
0x00	Network control octet
	Bit7 = 0 "BACnet APDU"
	Bit6 = 0 not used
	Bit5 = 0 DNET not present
	Bit4 = 0 not used
	Bit3 = 0 SNET not present
	Bit2 = 0 no reply expected
	Bit1,0 = 00 normal priority
	APDU
0x60	APDU type 6: Reject
0x06	Original Invoke ID 6
0x04	Reject Reason 4 (invalid-tag)
	MS/TP footer
0x47	Data CRC-16
0x41	

Abort APDUs

If a RECEIVER cannot acquire sufficient internal resources to service a request, or if the request contains a non-initial segment, the RECEIVER will return a BACnet-Abort-PDU. Since the RECEIVER is acting as a server for the request, it will set the "server" bit in the Abort PDU. The Abort PDU is defined in clause 20.1.9:

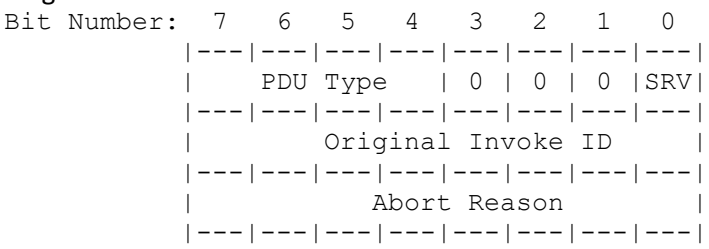
BACnet-Abort-PDU ::= SEQUENCE {

pdu-type	[0] Unsigned (0..15), -- 7 for this PDU type
reserved	[1] Unsigned (0..7), -- shall be set to zero
server	[2] BOOLEAN,
original-invoke-id	[3] Unsigned (0..255),
abort-reason	[4] BACnetAbortReason

-- Context specific tags 0..4 are NOT used in header encoding

}

The format of the BACnet-Abort-PDU is defined in clause 20.1.9.4, and is always three octets long.



The BACnetAbortReason enumeration is defined in Clause 21.

In this example, the abort reason will be "invalid-APDU-in-this-state":

	MS/TP header
0x55	Preamble
0xFF	Preamble
0x06	Frame Type: BACnet Data Not Expecting Reply (6)
0x01	Destination station address (unicast)
0x03	Source station address
0x00	Data Length = 5
0x05	
0xCA	Header CRC
	NPDU
0x01	Network version 1
0x00	Network control octet
	Bit7 = 0 "BACnet APDU"
	Bit6 = 0 not used
	Bit5 = 0 DNET not present
	Bit4 = 0 not used
	Bit3 = 0 SNET not present
	Bit2 = 0 no reply expected
	Bit1,0 = 00 normal priority
	APDU
0x71	APDU type 7: Abort, sent by server
0x06	Original Invoke ID 6
0x02	Abort Reason 2 (invalid-APDU-in-this-state)
	MS/TP footer
0x47	Data CRC-16
0x41	

Legal Stuff

This paper represents the author's opinions only and does not necessarily represent the views of the author's employer, SSPC-135, ASHRAE or any other organization. While the author believes all information is factual, it is provided as-is without any guarantees of suitability for a particular purpose. The name "BACnet" and its logo are registered trademarks of ASHRAE Inc.

Contact the Author

Steve Karg

Legrand, North America
1972 Waterford Place
Birmingham AL 35244 USA

+1-770-262-3095 voice
steve@kargs.net