BACnet
Application Layer Protocol

Lehrgangsunterlagen des AUTnet e. V. Erfurt
Teil E – 3
(Englisch)

Autor: Prof. Dr. F. Tiersch

Contents

Literature

[1]   Building automation and control systems – Part 5: Data communication protocol
         (ISO/DIS 16 484-5: 2002); English version prEN ISO 16 484-5: 2002

3 BACnet Application Layer

3.1 BACnet Application Layer Model

An application program within a BACnet node that want to communicate with an application program on another BACnet node uses the application layer of the node primarily. To describe and illustrate the interaction between the application program and the application layer an application layer model may be used (figure 3-1).
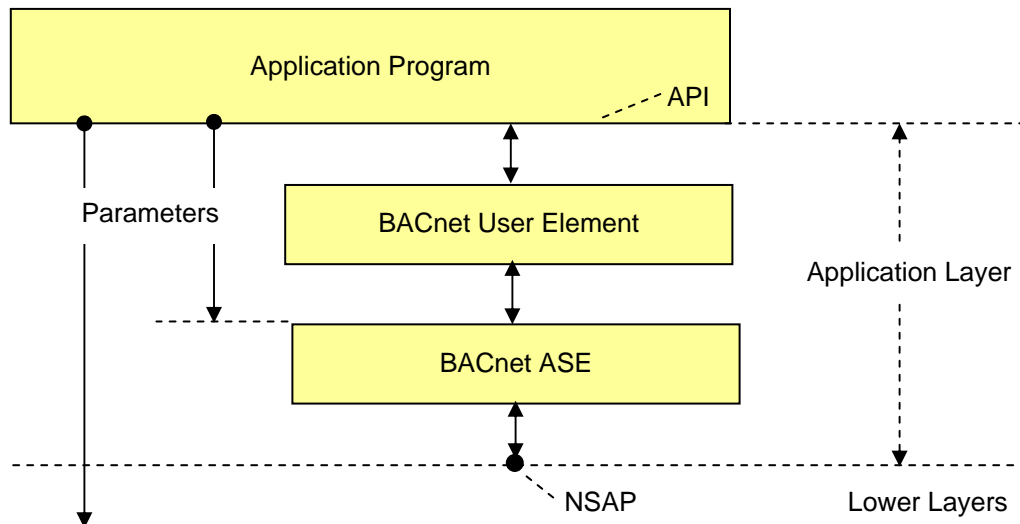

Figure 3-1: BACnet Application Layer Model (NSAP ~ Network Service Access Point) [1]

The application program interacts with the Application Layer through the Application Layer Program Interface (API). The BACnet Application Layer is itself made up of two parts. The BACnet User Element supports the local API and carries out several other functions. It implements the service procedure portion of an application service and maintains information about the context of a transaction. For instance, it generates the invokeIDs and remembers which invokeID goes with which application service request respectively response. It is also responsible for maintaining the time out counters that are required for the retrying of a transmission. The BACnet Application Service Element (ASE) represents a set of functions and application services for alarms, events, object accesses and so on.

3.2 BACnet Application Layer Services

Information exchange between two peer applications is represented in BACnet as an exchange of abstract service primitives (following the ISO conventions: ISO TR 8509). These primitives are used to convey service-specific parameters. Four service primitives are defined:

- request,
- indication,
- response and
- confirmation.

The information contained in the primitives is conveyed using a variety of protocol data units (PDUs). In BACnet, the following PDUs are used:

- CONF_SERV. request
- UNCONF_SERV.request
- SEGMENT_ACK.request
- REJECT.request
- ABORT.request

- CONF_SERV.response

- CONF_SERV.indication
- UNCONF_SERV. indication
- SEGMENT_ACK.indication
- REJECT.indication
- ABORT.indication

- CONF_SERV.confirm

The designation CONF_SERV. ... indicates that BACnet confirmed service PDUs are being used, and so on.

An application program that needs to communicate with a remote application process accesses the local BACnet User Element through the API (figures 3-1 and 3-2). Some of the API parameters is passed directly down to the network or data link layers. Such parameters may be the address of the device to which the service request is to send or protocol control information. The remainder of the parameters make up an application service primitive that passed from the BACnet User Element to the BACnet ASE.

Conceptionally, the application service primitive results in the generation of an APDU (figure 3-2) that becomes the data portion of a network service primitive, which is passed to the network layer through the Network Service Access Point (NSAP). Similarly this request passes down through the lower layers of the protocol stack in the local device. This process is illustrated in figure 3-2. The message is then transmitted to the remote device, where it is passed through the protocol stack and appears as an indication primitive in the application program of the remote device. In a similar fashion, the response from the remote device, if any, returns to the initiator of the service.
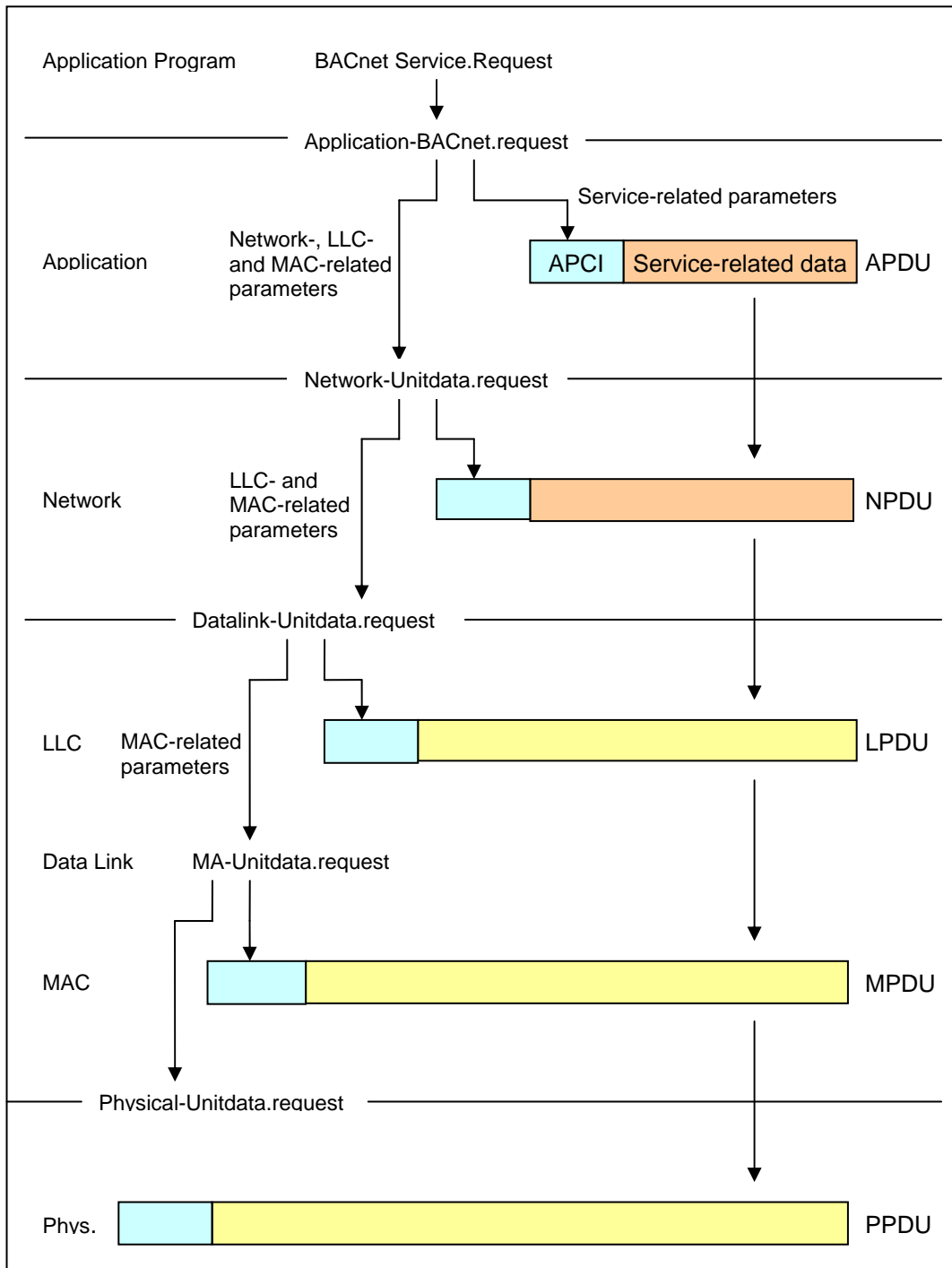
Figure 3-2: BACnet protocol stack and data flow [1] (gappy)

3.3   BACnet Application Protocol Sequence Diagrams

The flow sequence of service primitives can be represented by time-sequence diagrams. The figures 3-3 and 3-4 show time-sequence diagrams for the Normal Confirmed Service and the Normal Unconfirmed Service.
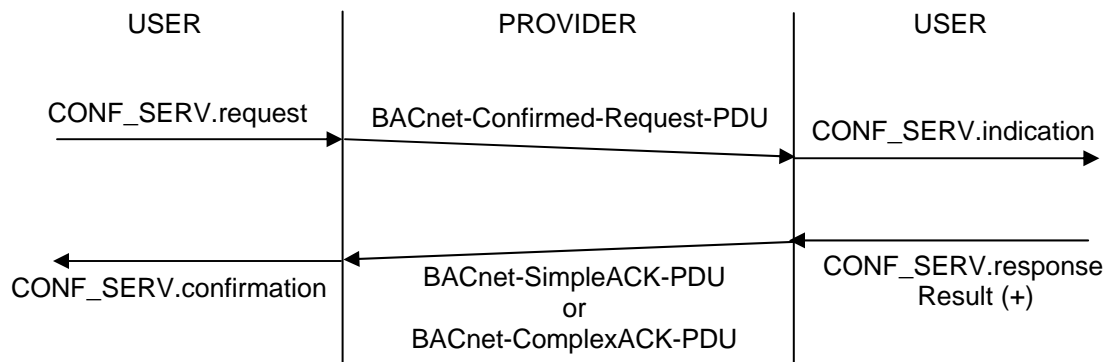
USER	PROVIDER	USER
CONF_SERV.request	BACnet-Confirmed-Request-PDU	CONF_SERV.indication
CONF_SERV.confirmation	BACnet-SimpleACK-PDU or BACnet-ComplexACK-PDU	CONF_SERV.response Result (+)

Figure 3-3: Time sequence diagram for a normal confirmed service [1]

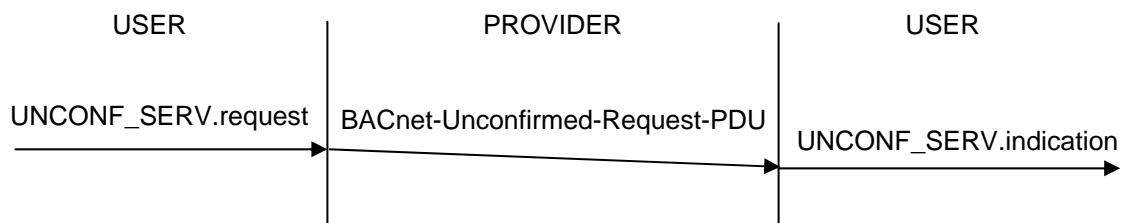| USER	PROVIDER	USER
UNCONF_SERV.request	BACnet-Unconfirmed-Request-PDU	UNCONF_SERV.indication

Figure 3-4: Time service diagram for an normal unconfirmed service [1]

Each diagram is partitioned into three fields. The fields labeld „User" represent the two service-users and the „Provider" field represents the service provider. The vertical lines between users and provider represent the interface between the BACnet User Element and the BACnet ASE.
For lower layers these vertical lines represent the service-access-points between the service-users and the service-provider. Arrows indicate the flow of information during an interaction described by a service-primitive. The figures illustrate sequences of application primitives defined in BACnet. Time sequence diagrams for abnormal services and services for segmented transactions are shown in the standard [1].

3.4   BACnet Application Service Conventions

The OSI conventions used in the BACnet standard define the interactions between a protocol service user and a service provider. Information passed between user and provider is represented abstractly as an exchange of „service primitives". The service primitives are an abstraction of

- the functional specification and
- the user-layer interaction.

Local details of the user/provider interaction are not contained.

A service primitive has a set of zero or more parameters that represent data elements qualifying the functions invoked by the primitive. These parameters may be explicitly stated or implicitely associated with the service access point.

BACnet uses a tabular format to describe the component parameters of the service primitives. Figure 3-5 shows the structure of such a service primitive.

Parameter-Name	Request	Indication	Response	Confirmation
Argument	M	M (=)		
Subscriber Process Identifier	M	M (=)		
Monitored Object	M	M (=)		
Issue ConfirmedNotifications	U	U (=)		
Lifetime	U	U (=)		
Result (+)			S	S (=)
Result (-)			S	S (=)
Error Type			M	M (=)

M ~ mandatory; U ~ user option; C ~ conditional upon other parameters; S ~ selection ot two or more possible parameters, parameters that make up this collection are identical in the table ; (=) ~ indicates that the parameter is semantically equivalent to the parameter in the service primitive to its immediate left in the table.

Figure 3-5: Structure of SubscribeCOV Service Primitives [1]

Some parameters may contain subparameters, indicated by intention. The presence of subparameters always depends on the presence of the parent parameters.

3.5   BACnet Application Protocol Data Units

The information contained in the application service primitives and the associated parameters are conveyed by using Application Layer Data Units (APDUs). BACnet's encoding rules for APDUs take into account the requirements of building automation and control systems. APDUs for use in building automation systems have to be simple and compactly.

The encoding rules taken in BACnet are a compromise between the encoding of ASN.1 (specified in ISO 8825) with its uniform encoding scheme of all data elements in a PDU and the mutual agreement as to its data format and location within the PDU in advance.

The fixed portion of each APDU containing protocol control information is encoded accordingly to ASN.1. Each data element is represented by tree components:

- identifier octets,

- length octets and
- contents octets.

The header, or fixed part, of the APDU comprises data required for the operation of the application layer protocol. It includes the

- type of APDU and
- information to carry out the reassembly of segmented messages.

The variable portion of each APDU containing service-specific information is encoded according to BACnet-specific rules. The resulting scheme significantly reduces overhead. It comprises information specific to individual service requests and responses, the „user data". Eight types of BACnet APDU types exist:

- BACnet-Unconfirmed-Request-PDU
- BACnet-Confirmed-Request-PDU
- BACnet-SimpleACK-PDU
- BACnet-ComplexACK-PDU
- BACnet-SegmentedACK-PDU
- BACnet-Abort-PDU
- BACnet-Error-PDU
- BACnet-Reject-PDU

The APDU type is encoded as a four-bit binary number in the bits 4 through 7 of the first octet of the APDU header, with bit 7 being the most significant bit.

Three examples show the encoding rules for APDUs in BACnet.

3.6   Example 1: BACnet-Confirmed-Request-PDU

The BACnet-Confirmed-Request-PDU is used to convey the information contained in confirmed service request primitives. It is encoded as B'0000' in the first octet (see Figure 3-6).

The following parameters refer to segmentation of the message. Segmentation is not used in our example. But B'0010' means that the device issuing the confirmed request will accept a segmented complex acknowledgement as a response.

Bits 0 ... 3 of the second octet specify the maximum size of a single APDU that the issuing device will accept. B'0101' allows up to 1476 octets that fit in an ISO 8802-3 frame.

The parameter „InvokeID" is used to associate the response to a confirmed service request with the original request. The „InvokeID" shall be returned by the service provider. This may be done in a BACnet-SimpleACK-PDU or a BACnet-ComplexACK-PDU. In case of an error condition, the „InvokeID" shall be returned in an Error-, Reject- or Abort-PDU. The „InvokeID", generated by the requesting device, shall be unique for all outstanding confirmed request APDUs generated by this device. It shall be maintained within the device until either a response APDU is received or a no response timer expires. The „InvokeID" is only source-device-unique and therefore the responding device shall maintain it as well as the address of the requesting device until a response has been sent.
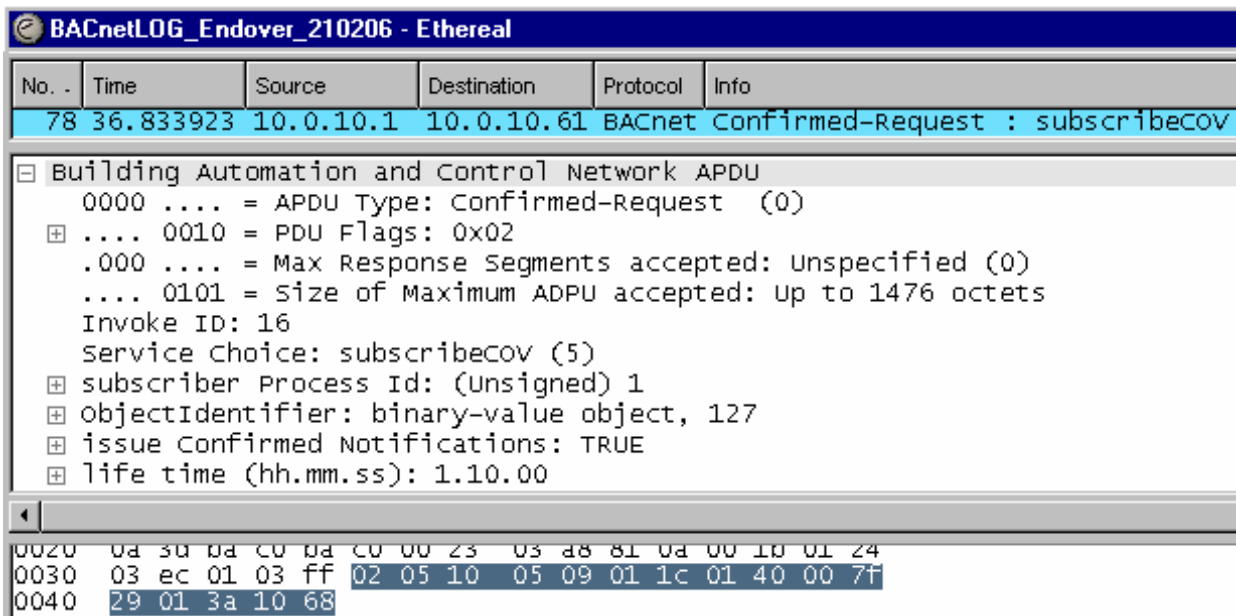

Figure 3-6: BACnet-APDU-part in a frame of a real BACnet Building Automation System

Figure 3-7: Format of the BACnet-Confirmed-Request-PDU [1] and the values of Example 1 (gappy)

The parameter „Service Choice" contains the value of the BACnetConfirmedServiceChoice, in the example of Figure 3-6 the value „5". This means „SubscribeCOV Service", one of the confirmed Alarm and Event Services. Figure 3-7 shows the format of the BACnet-Confirmed-Request-PDU.

The following parameters correspond the variable portion of the APDU. They are also parameters of the SubscribeCOV Service and shall be described there.

Change of value (COV) reporting is one of three mechanisms for managing events. The others are intrinsic reporting and algorithmic change reporting. "Events are changes of value of certain properties of certain objects, or internal status changes, that meet predetermined criteria" [1].
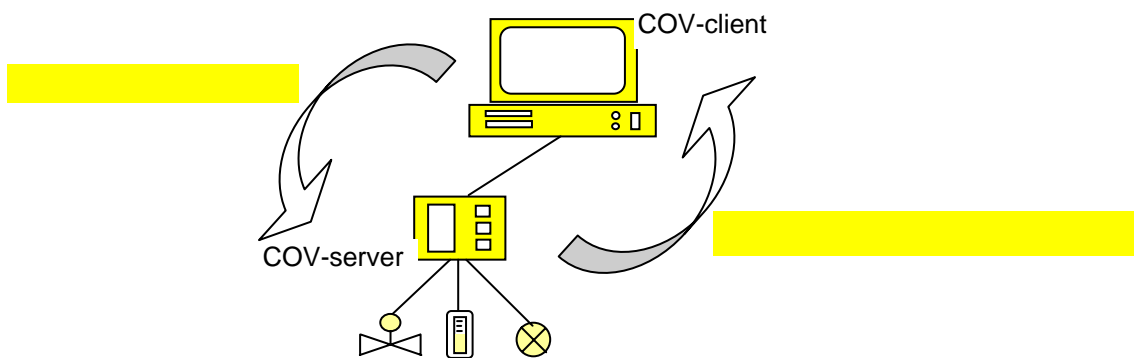

Figure 3-8: SubscribeCOV and ConfirmedCOVNotification services (gappy)

SubscribeCOV service is used by a COV-client to subscribe for the receipt of notifications of changes. A subscription establishes a connection between the mechanism for change of value detection and reporting within a COV-server and an application process within the COV-client device. A change of value of specific properties of an object, supporting COV reporting, triggers COV notification to be sent to one or more subscriber clients, typically to supervisory programs in BACnet client devices or to operators or logging devices. The COV-server device can issue a notification of a change after the subscription has been established. For this purpose the COV-server device uses the ConfirmedCOVNotification service or the UnconfirmedCOVNotification service, choosen at the time the subscription has been established (figure 3-8). The structure of the SubscribeCOV service primitive is shown in figure 3-9.


```
    Service Choice: subscribeCOV (5)
⊞ subscriber Process Id: (Unsigned) 1
⊞ ObjectIdentifier: binary-value object, 127
⊞ issue Confirmed Notifications: TRUE
⊞ life time (hh.mm.ss): 1.10.00
```

Figure 3-9: Extract of Figure 3-6

The „Subscriber Process Identifier" conveys a numeric handle to the subscriber to match future re-subscriptions or cancellations. In our example, the handle is „1". It shall be later used with ConfirmedCOVNotification service to identify the process within the COV-client that should receive them.

The parameter „Monitored Object Identifier" shall be used to convey the object-identifier (in our example „127" of the object type „binary-value") within the receiving device for which a subscription is desired. With the parameter „Issue Confirmed Notification" shall be defined whether the COV-server device shall issue Confirmed or UnconfirmedCOVNotifications when changes occur.
The parameter „Lifetime" conveys the desired lifetime of the subscription in seconds. After this time the subscription shall be automatically cancelled. If both the parameters „Issue ..." and „Lifetime" are absent, then this indicates a cancellation request.