```
X'02000009'             Device, Instance Number=9
X'2C'                   SD Context Tag 2 (Event Object Identifier, L=4)
X'00000002'             Analog Input, Instance Number=2
X'3E'                   PD Opening Tag 3 (Time Stamp)
    X'19'                   SD Context Tag 1 (SequenceNumber, L=1)
    X'10'                   16
X'3F'                   PD Closing Tag 3 (Time Stamp)
X'49'                   SD Context Tag 4 (Notification Class, L=1)
X'04'                   4
X'59'                   SD Context Tag 5 (Priority, L=1)
 X'64'                  100
X'69'                   SD Context Tag 6 (Event Type, L=1)
X'05'                   5 (OUT_OF_RANGE)
X'89'                   SD Context Tag 8 (Notify Type, L=1)
X'00'                   0 (ALARM)
X'99'                   SD Context Tag 9 (AckRequired, L=1)
X'01'                   1 (TRUE)
X'A9'                   SD Context Tag 10 (From State, L=1)
X'00'                   0 (NORMAL)
X'B9'                   SD Context Tag 11 (To State, L=1)
X'03'                   3 (HIGH_LIMIT)
X'CE'                   PD Opening Tag 12 (Event Values)
    X'5E'                   PD Opening Tag 5 (BACnetNotificationParameters, Choice 5=OUT_OF_RANGE)
        X'0C'                   SD Context Tag 0 (Exceeding Value, L=4)
        X'42A03333'             80.1
        X'1A'                   SD Context Tag 1 (Status Flags, L=2)
        X'0480'                 1,0,0,0 (TRUE,FALSE,FALSE,FALSE)
        X'2C'                   SD Context Tag 2 (Deadband, L=4)
        X'3F800000'             1.0
        X'3C'                   SD Context Tag 3 (Exceeded Limit, L=4)
        X'42A00000'             80.0
    X'5F'                   PD Closing Tag 5 (BACnetNotificationParameters)
X'CF'                   PD Closing Tag 12 (Event Values)
```

At some future time, an AcknowledgeAlarm-Request is generated:

```
X'00'                   PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X'02'                   Maximum APDU Size Accepted=206 octets
X'07'                   Invoke ID=7
X'00'                   Service Choice=0 (AcknowledgeAlarm-Request)

X'09'                   SD Context Tag 0 (Acknowledging Process Identifier, L=1)
X'01'                   Acknowledging Process Identifier=1
X'1C'                   SD Context Tag 1 (Event Object Identifier, L=4)
X'00000002'             Analog Input, Instance Number=2
X'29'                   SD Context Tag 2 (Event State Acknowledged, L=1)
X'03'                   3 (HIGH_LIMIT)
X'3E'                   PD Opening Tag 3 (Time Stamp)
    X'19'                   SD Context Tag 1 (Sequence Number, L=1)
    X'10'                   16
X'3F'                   PD Closing Tag 3 (Time Stamp)
X'4C'                   SD Context Tag 4 (Acknowledgment Source, L=4)
X'00'                   ANSI X3.4 Encoding
X'4D444C'               "MDL"
X'5E'                   PD Opening Tag 5 (Time Of Acknowledgment)
```

```
        X'2E'                   PD Opening Tag 2 (Date Time)
            X'A4'               Application Tag 10 (Date, L=4)
            X'5C0615FF'         June 21, 1992
            X'B4'               Application Tag 11 (Time, L=4)
            X'0D032909'         13:03:41.9
        X'2F'                   PD Closing Tag 2 (Date Time)
X'5F'                   PD Closing Tag 5 (Time Of Acknowledgment)
```

Assuming the service procedure executes correctly, a simple acknowledgement is returned:

```
X'20'                   PDU Type=2 (BACnet-SimpleACK-PDU)
X'07'                   Invoke ID=7
X'00'                   Service ACK Choice=0 (AcknowledgeAlarm)
```

F.2 Example Encodings for File Access Services

F.2.1 Encoding for Example E.2.1 - AtomicReadFile Service

Example 1: Read data from a file.

```
X'00'                   PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X'02'                   Maximum APDU Size Accepted=206 octets
X'00'                   Invoke ID=0
X'06'                   Service Choice=6 (AtomicReadFile-Request)

X'C4'                   Application Tag 12 (Object Identifier, L=4) (File Identifier)
X'02800001'             File, Instance Number=1
X'0E'                   PD Opening Tag 0 (Stream Access)
        X'31'               Application Tag 3 (Signed Integer, L=1) (File Start Position)
        X'00'               0
        X'21'               Application Tag 2 (Unsigned, L=1) (Requested Octet Count)
        X'1B'               27
X'0F'                   PD Closing Tag 0 (Stream Access)
```

Assuming this service procedure executes correctly, a complex acknowledgment is returned:

```
X'30'                   PDU Type=3 (BACnet-ComplexACK-PDU, SEG=0, MOR=0)
X'00'                   Invoke ID=0
X'06'                   Service ACK Choice=6 (AtomicReadFile-ACK)

X'10'                   Application Tag 1 (Boolean, 0 (FALSE)) (End Of File)
X'0E'                   PD Opening Tag 0 (Stream Access)
        X'31'               Application Tag 3 (Signed Integer, L=1) (File Start Position)
        X'00'               0
        X'65'               Application Tag 6 (Octet String, L>4)
        X'1B'               Extended Length=27
        X'4368696C6C65723031204F6E2D54696D653D342E3320486F757273'
                            "Chiller01 On-Time=4.3 Hours"
X'0F'                   PD Closing Tag 0 (Stream Access)
```

Example 2: Read record from a file.

```
X'00'                   PDU type=0 (BACnet-Confirmed-Request-PDU,  SEG=0, MOR=0, SA=0)
X'02'                   Maximum APDU Size Accepted=206 octets
 X'12'                  Invoke ID=18
```

```
X'06'                   Service Request=6 (AtomicReadFile-Request)

X'C4'                   Application Tag 12 (Object Identifier, L=4) (FileIdentifier)
X'02800002'             File, Instance Number=2
X'1E'                   PD Opening Tag 1 (Record Access)
      X'31'                   Application Tag 3 (Signed Integer, L=1) (File Start Record)
      X'0E'                   14
      X'21'                   Application Tag 2 (Unsigned, L=1) (Requested Record Count)
      X'03'                   3
X'1F'                   PD Closing Tag 1 (Record Access)
```

Assuming this service procedure executes correctly, a complex acknowledgement is returned:

```
X'30'                   PDU Type=3 (BACnet-ComplexACK-PDU,  SEG=0, MOR=0)
X'12'                   Invoke ID=18
X'06'                   Service ACK Choice=6 (AtomicReadFile-ACK)

X'11'                   Application Tag 1 (Boolean, 1 (TRUE)) (End Of File)
X'1E'                   PD Opening Tag 1(RecordAccess)
      X'31'                   Application Tag 3 (Signed Integer, L=1) (File Start Record)
      X'0E'                   14
      X'21'                   Application Tag 2 (Unsigned, L=1) (Returned Record Count)
      X'02'                   2
      X'65'                   Application Tag 6 (Octet String, L>4) (File Record Data)
      X'0A'                   Extended Length=10
      X'31323A30302C34352E36' "12:00,45.6"'
      X'65'                   Application Tag 6 (Octet String, L>4) (File Record Data)
      X'0A'                   Extended Length=10
      X'31323A31352C34342E38' "12:15,44.8"
X'1F'                   PD Closing Tag 1 (Record Access)
```

F.2.2 Encoding for Example E.2.2 - AtomicWriteFile Service

Example 1: Write data to a file.

```
X'00'                   PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X'02'                   Maximum APDU Size Accepted=206 octets
X'55'                   Invoke ID=85
X'07'                   Service Choice=7 (AtomicWriteFile-Request)

X'C4'                   Application Tag 12 (Object Identifier, L=4) (File Identifier)
X'02800001'             File, Instance Number=1
X'0E'                   PD Opening Tag 0 (Stream Access)
      X'31'                   Application Tag 3 (Signed Integer, L=1) (File Start Position)
      X'1E'                   30
      X'65'                   Application Tag 6 (Octet String, L>4) (File Data)
      X'1B'                   Extended Length=27
      X'4368696C6C65723031204F6E2D54696D653D342E3320486F757273'
                              "Chiller01 On-Time=4.3 Hours"
X'0F'                   PD Closing Tag 0 (Stream Access)
```

Assuming this service procedure executes correctly, a complex acknowledgement is returned:

```
X'30'                   PDU Type=3 (BACnet-ComplexACK-PDU, SEG=0, MOR=0)
X'55'                   Invoke ID=85
```

```
X'07'                   Service ACK Choice=7 (AtomicWriteFile-ACK)
X'09'                   SD Context Tag 0 (File Start Position, L=1)
X'1E'                   30
```

Example 2: Append two records to a file.

```
X'00'                   PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X'02'                   Maximum APDU Size Accepted=206 octets
X'55'                   Invoke ID=85
X'07'                   Service Choice=7 (AtomicWriteFile-Request)

X'C4'                   Application Tag 12 (Object Identifier, L=4) (File Identifier)
X'02800002'             File, Instance Number=2
X'1E'                   PD Opening Tag 1 (Record Access)
        X'31'                   Application Tag 3 (Signed Integer, L=1) (File Start Record)
        X'FF'                   -1 (Append to End of File)
        X'21'                   Application Tag 2 (Unsigned Integer, L=1) (Record Count)
        X'02'                   2
        X'65'                   Application Tag 6 (Octet String, L>4) (File Record Data)
        X'0A'                   Extended Length=10
        X'31323A30302C34352E36' "12:00,45.6"
        X'65'                   Application Tag 6 (Octet String, L>4) (File Record Data)
        X'0A'                   Extended Length=10
        X'31323A31352C34342E38' "12:15,44.8"
X'1F'                   PD Closing Tag 1 (Record Access)
```

Assuming this service procedure executes correctly, a complex acknowledgement is returned:

```
X'30'                   PDU Type=3 (BACnet-ComplexACK-PDU, SEG=0, MOR=0)
X'55'                   Invoke ID=85
X'07'                   Service ACK Choice=7 (AtomicWriteFile-ACK)
X'19'                   SD Context Tag 1 (File Start Record, L=1)
X'0E'                   14
```

F.3 Example Encodings for Object Access Services

F.3.1 Encoding for Example E.3.1 - AddListElement Service

```
X'00'                   PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X'02'                   Maximum APDU Size Accepted=206 octets
X'01'                   Invoke ID=1
X'08'                   Service Choice=8 (AddListElement-Request)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'02C00003'             Group, Instance Number=3
X'19'                   SD Context Tag 1 (Property Identifier, L=1)
X'35'                   53 (LIST_OF_GROUP_MEMBERS)
X'3E'                   PD Opening Tag 3 (List Of Elements)
        X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
        X'0000000F'             Analog Input, Instance Number=15
        X'1E'                   PD Opening Tag 1 (List Of Property References)
                X'09'                   SD Context Tag 0 (Property Identifier, L=1)
                X'55'                   85 (PRESENT_VALUE)
                X'09'                   SD Context Tag 0 (Property Identifier, L=1)
                X'67'                   103 (RELIABILITY)
```

```
        X'1F'                   PD Closing Tag 1 (List Of Property References)
X'3F'                   PD Closing Tag 3 (List Of Elements)
```

Assuming this service procedure executes correctly, a simple acknowledgement is returned:

```
X'20'                   PDU Type=2 (BACnet-SimpleACK-PDU)
X'01'                   Invoke ID=1
X'08'                   Service ACK Choice=8 (AddListElement)
```

F.3.2 Encoding for Example E.3.2 - RemoveListElement Service

```
X'00'                   PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X'02'                   Maximum APDU Size Accepted=206 octets
X'34'                   Invoke ID=52
X'09'                   Service Choice=9 (RemoveListElement-Request)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'02C00003'             Group, Instance Number=3
X'19'                   SD Context Tag 1 (Property Identifier, L=1)
X'35'                   53 (LIST_OF_GROUP_MEMBERS)
X'3E'                   PD Opening  Tag 3 (List Of Elements)
        X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
        X'0000000C'             Analog Input, Instance Number=12
        X'1E'                   PD Opening Tag 1 (List Of Property References)
                X'09'                   SD Context Tag 0 (Property Identifier, L=1)
                X'55'                   85 (PRESENT_VALUE)
                X'09'                   SD Context Tag 0 (Property Identifier, L=1)
                X'67'                   103 (RELIABILITY)
                X'09'                   SD Context Tag 0 (Property Identifier, L=1)
                X'1C'                   28 (DESCRIPTION)
        X'1F'                   PD Closing Tag 1 (List Of Property References)

        X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
        X'0000000D'             Analog Input, Instance Number=13
        X'1E'                   PD Opening Tag 1 (List Of Property References)
                X'09'                   SD Context Tag 0 (Property Identifier, L=1)
                X'55'                   85 (PRESENT_VALUE)
                X'09'                   SD Context Tag 0 (Property Identifier, L=1)
                X'67'                   103 (RELIABILITY)
                X'09'                   SD Context Tag 0 (Property Identifier, L=1)
                X'1C'                   28 (DESCRIPTION)
        X'1F'                   PD Closing Tag 1 (List Of Property References)
X'3F'                   PD Closing Tag 3 (List Of Elements)
```

Assuming the service procedure executes correctly, a simple acknowledgment is returned:

```
X'20'                   PDU Type=2 (BACnet-SimpleACK-PDU)
X'34'                   Invoke ID=52
X'09'                   Service ACK Choice=9 (RemoveListElement)
```

This second part of the example re-inserts two of the three elements removed above:

```
X'00'                   PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X'02'                   Maximum APDU Size Accepted=206 octets
X'35'                   Invoke ID=53
```

```
X'08'                   Service Choice=8 (AddListElement-Request)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'02C00003'             Group, Instance Number=3
X'19'                   SD Context Tag 1 (Property Identifier, L=1)
X'35'                   53 (LIST_OF_GROUP_MEMBERS)
X'3E'                   PD Opening  Tag 3 (List Of Elements)
      X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
      X'0000000C'             Analog Input, Instance Number=12
      X'1E'                   PD Opening Tag 1 (List Of Property References)
            X'09'                   SD Context Tag 0 (Property Identifier, L=1)
            X'55'                   85 (PRESENT_VALUE)
            X'09'                   SD Context Tag 0 (Property Identifier, L=1)
            X'67'                   103 (RELIABILITY)
      X'1F'                   PD Closing Tag 1 (List Of Property References)

      X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
      X'0000000D'             Analog Input, Instance Number=13
      X'1E'                   PD Opening Tag 1 (List Of Property References)
            X'09'                   SD Context Tag 0 (Property Identifier, L=1)
            X'55'                   85 (PRESENT_VALUE)
            X'09'                   SD Context Tag 0 (Property Identifier, L=1)
            X'67'                   103 (RELIABILITY)
      X'1F'                   PD Closing Tag 1 (List Of Property References)
X'3F'                   PD Closing Tag 3 (List Of Elements)
```

Assuming the service procedure executes correctly, a simple acknowledgment is returned:

```
X'20'                   PDU Type=2 (BACnet-SimpleACK-PDU)
X'35'                   Invoke ID=53
X'08'                   Service ACK Choice=8 (AddListElement)
```

F.3.3 Encoding for Example E.3.3 - CreateObject Service

```
X'00'                   PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X'04'                   Maximum APDU Size Accepted=1024 octets
X'56'                   Invoke ID=86
X'0A'                   Service Choice=10 (CreateObject-Request)

X'0E'                   PD Opening Tag 0 (Object Specifier)
      X'09'                   SD Context Tag 0 (Object Type, L=1)
      X'0A'                   10 (File Object)
X'0F'                   PD Closing Tag 0 (Object Specifier)
X'1E'                   PD Opening Tag 1 (List Of Initial Values)
      X'09'                   SD Context Tag 0 (Property Identifier, L=1)
      X'4D'                   77 (OBJECT_NAME)
      X'2E'                   PD Opening Tag 2 (Value)
            X'75'                   Application Tag 7 (Character String, L>4)
            X'08'                   Extended Length=8
            X'00'                   ANSI X3.4 Encoding
            X'5472656E642031'  "Trend 1"
      X'2F'                   PD Closing Tag 2 (Value)
```

```
        X'09'                SD Context Tag 0 (Property Identifier, L=1)
        X'29'                41 (FILE_ACCESS_METHOD)
        X'2E'                PD Opening Tag 2 (Value)
                X'91'            Application Tag 9 (Enumerated, L=1)
                X'00'            0 (RECORD_ACCESS)
        X'2F'                PD Closing Tag 2 (Value)
X'1F'                PD Closing Tag 1 (List Of Initial Values)
```

Assuming the service procedure executes correctly, an acknowledgement is returned conveying the new object identifier:

```
X'30'           PDU Type=3 (BACnet-ComplexACK-PDU, SEG=0, MOR=0)
X'56'           Invoke ID=86
X'0A'           Service ACK Choice=10 (CreateObject-ACK)
X'C4'           Application Tag 12 (Object Identifier, L=4)
X'0280000D'     File, Instance Number=13
```

F.3.4 Encoding for Example E.3.4 - DeleteObject Service

Example 1: A successful attempt to delete an object.

```
X'00'           PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X'04'           Maximum APDU Size Accepted=1024 octets
X'57'           Invoke ID=87
X'0B'           Service Choice=11 (DeleteObject-Request)
X'C4'           Application Tag 12 (Object Identifier, L=4)
X'02C00006'     Group, Instance Number=6
```

Assuming the service procedure executes correctly, a simple acknowledgement is returned:

```
X'20'           PDU Type=2 (BACnet-SimpleACK-PDU)
X'57'           Invoke ID=87
X'0B'           Service ACK Choice=11 (DeleteObject)
```

Example 2: An unsuccessful attempt to delete an object.

```
X'00'           PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X'04'           Maximum APDU Size Accepted=1024 octets
X'58'           Invoke ID=88
X'0B'           Service Choice=11 (DeleteObject-Request)

X'C4'           Application Tag 12 (Object Identifier, L=4)
X'02C00007'     Group, Instance Number=7
```

In this example, the object is assumed to be protected and cannot be deleted by this protocol service. The server issues the following response:

```
X'50'           PDU Type=5 (BACnet-Error-PDU)
X'58'           Original Invoke ID=88
X'0B'           Error Choice=11 (DeleteObject)
X'91'           Application Tag 9 (Enumerated, L=1) (Error Class)
X'01'           1 (OBJECT)
X'91'           Application Tag 9 (Enumerated, L=1) (Error Code)
X'17'           23 (OBJECT_DELETION_NOT_PERMITTED)
```

F.3.5 Encoding for Example E.3.5 - ReadProperty Service

```
X'00'                   PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X'00'                   Maximum APDU Size Accepted=50 octets
X'01'                   Invoke ID=1
X'0C'                   Service Choice=12 (ReadProperty-Request)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'00000005'             Analog Input, Instance Number=5
X'19'                   SD Context Tag 1 (Property Identifier, L=1)
X'55'                   85 (PRESENT_VALUE)
```

This request produces the following result:

```
X'30'                   PDU Type=3 (BACnet-ComplexACK-PDU, SEG=0, MOR=0)
X'01'                   Invoke ID=1
X'0C'                   Service ACK Choice=12 (ReadProperty-ACK)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'00000005'             Analog Input, Instance Number=5
X'19'                   SD Context Tag 1 (Property Identifier, L=1)
X'55'                   85 (PRESENT_VALUE)
X'3E'                   PD Opening Tag 3 (Property Value)
    X'44'                   Application Tag 4 (Real, L=4)
    X'4290999A'             72.3
X'3F'                   PD Closing Tag 3 (Property Value)
```

F.3.6 Encoding for Example E.3.6 - ReadPropertyConditional Service

Example 1: Getting the object identifiers of all Binary Input objects.

```
X'00'                   PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X'04'                   Maximum APDU Size Accepted=1024 octets
X'51'                   Invoke ID=81
X'0D'                   Service Choice=13 (ReadPropertyConditional Request)

X'0E'                   PD Opening Tag 0 (Object Selection Criteria)
    X'09'                   SD Context Tag 0 (Selection Logic, L=1)
    X'00'                   0 (AND)
    X'1E'                   PD Opening Tag 1 (List Of Selection Criteria)
        X'09'                   SD Context Tag 0 (Property Identifier, L=1)
        X'4F'                   79 (OBJECT_TYPE)
        X'29'                   SD Context Tag 2 (Relation Specifier)
        X'00'                   0 (EQUAL)
        X'3E'                   PD Opening Tag 3 (Comparison Value)
            X'91'                   Application Tag 9 (Enumerated, L=1) (Object Type)
            X'03'                   3 (BINARY_INPUT)
        X'3F'                   PD Closing Tag 3 (Comparison Value)
    X'1F'                   PD Closing Tag 1 (List Of Selection Criteria)
X'0F'                   PD Closing Tag 0 (Object Selection Criteria)
```

This request produces the following result:

```
X'30'                   PDU Type=3 (BACnet-ComplexACK-PDU, SEG=0, MOR=0)
X'51'                   Invoke ID=81
X'0D'                   Service Ack Choice=13 (ReadPropertyConditional-ACK)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'00C00001'             Binary Input, Instance Number=1

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'00C00002'             Binary Input, Instance Number=2

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'00C00003'             Binary Input, Instance Number=3

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'00C00004'             Binary Input, Instance Number=4
```

Example 2: Conditional read of Analog Inputs.

```
X'00'                   PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X'04'                   Maximum APDU Size Accepted=1024 octets
X'52'                   Invoke ID=82
X'0D'                   Service Choice=13 (ReadPropertyConditional Request)

X'0E'                   PD Opening Tag 0 (Object Selection Criteria)
    X'09'                   SD Context Tag 0 (Selection Logic, L=1)
    X'00'                   Enumeration 0 (AND)
    X'1E'                   PD Opening Tag 1 (List Of Selection Criteria)
        X'09'                   SD Context Tag 0 (Property Identifier, L=1)
        X'4F'                   79 (OBJECT_TYPE)
        X'29'                   SD Context Tag 2 (Relation Specifier)
        X'00'                   0 (EQUAL)
        X'3E'                   PD Opening Tag 3 (Comparison Value)
            X'91'                   Application Tag 9 (Enumerated, L=1) (Object Type)
            X'00'                   0 (ANALOG_INPUT)
        X'3F'                   PD Closing Tag 3 (Comparison Value)

        X'09'                   SD Context Tag 0 (Property Identifier, L=1)
        X'55'                   85 (PRESENT_VALUE)
        X'29'                   SD Context Tag 2 (Relation Specifier)
        X'03'                   3 (GREATER_THAN)
        X'3E'                   PD Opening  Tag 3 (Comparison Value)
            X'44'                   Application Tag 4 (Real, L=4)
            X'42C80000'                 100.0
        X'3F'                   PD Closing  Tag 3 (Comparison Value)
    X'1F'                   PD Closing Tag 1 (List Of Selection Criteria)
X'0F'                   PD Closing Tag 0 (Object Selection Criteria)
```

This request produces the following result:

```
X'30'                   PDU Type=3 (BACnet-ComplexACK-PDU, SEG=0, MOR=0)
X'52'                   Invoke ID=82
```

```
X'0D'                   Service Ack Choice=13 (ReadPropertyConditional-ACK)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'00000001'             Analog Input, Instance Number=1
X'1E'                   PD Opening Tag 1 (List Of Results)
    X'29'                   SD Context Tag 2 (Property Identifier, L=1)
    X'55'                   85 (PRESENT_VALUE)
    X'4E'                   PD Opening  Tag 4 (Property Value)
        X'44'                   Application Tag 4 (Real, L=4)
        X'42CC999A'             102.3
    X'4F'                   PD Closing Tag 4 (Property Value)
X'1F'                   PD Closing Tag 1 (List Of Results)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'00000002'             Analog Input, Instance Number=2
X'1E'                   PD Opening Tag 1 (List Of Results)
    X'29'                   SD Context Tag 2 (Property Identifier, L=1)
    X'55'                   85 (PRESENT_VALUE)
    X'4E'                   PD Opening  Tag 4 (Property Value)
        X'44'                   Application Tag 4 (Real, L=4)
        X'4334B333'             180.7
    X'4F'                   PD Closing  Tag 4 (Property Value)
X'1F'                   PD Closing Tag 1 (List Of Results)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'00000003'             Analog Input, Instance Number=3
X'1E'                   PD Opening Tag 1 (List Of Results)
    X'29'                   SD Context Tag 2 (Property Identifier, L=1)
    X'55'                   85 (PRESENT_VALUE)
    X'4E'                   PD Opening  Tag 4 (Property Value)
        X'44'                   Application Tag 4 (Real, L=4)
        X'430E199A'             142.1
    X'4F'                   PD Closing  Tag 4 (Property Value)
X'1F'                   PD Closing Tag 1 (List Of Results)
```

Example 3: Finding unreliable or out-of-service objects.

```
X'00'                   PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X'04'                   Maximum APDU Size Accepted=1024 octets
X'53'                   Invoke ID=83
X'0D'                   Service Choice=13 (ReadPropertyConditional Request)

X'0E'                   PD Opening Tag 0 (Object Selection Criteria)
    X'09'                   SD Context Tag 0 (Selection Logic, L=1)
    X'01'                   1 (OR)
    X'1E'                   PD Opening Tag 1 (List Of Selection Criteria)
        X'09'                   SD Context Tag 0 (Property Identifier, L=1)
        X'67'                   103 (RELIABILITY)
        X'29'                   SD Context Tag 2 (Relation Specifier)
        X'01'                   1 (NOT_EQUAL)
        X'3E'                   PD Opening Tag 3 (Comparison Value)
            X'91'                   Application Tag 9 (Enumerated, L=1)
            X'00'                   0 (NO_FAULT_DETECTED)
        X'3F'                   PD Closing Tag 3 (Comparison Value)
```

```
                X'09'           SD Context Tag 0 (Property Identifier, L=1)
                X'51'           81 (OUT_OF_SERVICE)
                X'29'           SD Context Tag 2 (Relation Specifier)
                X'00'           0 (EQUAL)
                X'3E'           PD Opening Tag 3 (Comparison Value)
                    X'11'           Application Tag 1 (Boolean, 1 (TRUE))
                X'3F'           PD Closing Tag 3 (Comparison Value)
        X'1F'               PD Closing Tag 1 (List Of Selection Criteria)
X'0F'                   PD Closing Tag 0 (Object Selection Criteria)

X'1E'                   PD Opening Tag 1 (List Of Property References)
        X'09'               SD Context Tag 0 (Property Identifier, L=1)
        X'55'               85 (PRESENT_VALUE)
        X'09'               SD Context Tag 0 (Property Identifier, L=1)
        X'67'               103 (RELIABILITY)
        X'09'               SD Context Tag 0 (Property Identifier, L=1)
        X'51'               81 (OUT_OF_SERVICE)
X'1F'                   PD Closing Tag 1 (List Of Property References)
```

Assuming the service procedure executes correctly, a complex acknowledgement is returned containing the requested values:

```
X'30'                   PDU Type=3 (BACnet-ComplexACK-PDU, SEG=0, MOR=0)
X'53'                   Invoke ID=83
X'0D'                   Service Ack Choice=13 (ReadPropertyConditional-ACK)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'00C00001'             Binary Input, Instance Number=1
X'1E'                   PD Opening Tag 1 (List Of Results)
        X'29'               SD Context Tag 2 (Property Identifier, L=1)
        X'55'               85 (PRESENT_VALUE)
        X'4E'               PD Opening Tag 4 (Property Value)
                X'91'           Application Tag 9 (Enumerated, L=1)
                X'01'           1 (ACTIVE)
        X'4F'               PD Closing Tag 4 (Property Value)

        X'29'               SD Context Tag 2 (Property Identifier, L=1)
        X'67'               103 (RELIABILITY)
        X'4E'               PD Opening Tag 4 (Property Value)
                X'91'           Application Tag 9 (Enumerated, L=1)
                X'00'           0 (NO_FAULT_DETECTED)
        X'4F'               PD Closing Tag 4 (Property Value)

        X'29'               SD Context Tag 2 (Property Identifier, L=1)
        X'51'               81 (OUT_OF_SERVICE)
        X'4E'               PD Opening Tag 4 (Property Value)
                X'11'           Application Tag 1 (Boolean, 1 (TRUE))
        X'4F'               PD Closing Tag 4 (Property Value)
X'1F'                   PD Closing Tag 1 (List Of Results)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'00000002'             Analog Input, Instance Number=2
X'1E'                   PD Opening Tag 1 (List Of Results)
        X'29'               SD Context Tag 2 (Property Identifier, L=1)
        X'55'               85 (PRESENT_VALUE)
        X'4E'               PD Opening Tag 4 (Property Value)
```

```
                X'44'               Application Tag 4 (Real, L=4)
                X'437A0000'         250.0
        X'4F'                   PD Closing Tag 4 (Property Value)

        X'29'                   SD Context Tag 2 (Property Identifier, L=1)
        X'67'                   103 (RELIABILITY)
        X'4E'                   PD Opening Tag 4 (Property Value)
                X'91'               Application Tag 9 (Enumerated, L=1)
                X'07'               7 (UNRELIABLE_OTHER)
        X'4F'                   PD Closing Tag 4 (Property Value)

        X'29'                   SD Context Tag 2 (Property Identifier, L=1)
        X'51'                   81 (OUT_OF_SERVICE)
        X'4E'                   PD Opening Tag 4 (Property Value)
                X'10'               Application Tag 1 (Boolean, 0 (FALSE))
        X'4F'                   PD Closing Tag 4 (Property Value)
X'1F'                       PD Closing Tag 1 (List Of Results)
```

Example 4: Finding the identifiers of all objects with Object_Names that match the comparision values "C* Pressure" and "AC? Supply Temp".

```
X'00'                       PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X'04'                       Maximum APDU Size Accepted=1024 octets
X'54'                       Invoke ID=84
X'0D'                       Service Choice=13 (ReadPropertyConditional Request)

X'0E'                       PD Opening Tag 0 (Object Selection Criteria)
        X'09'                   SD Context Tag 0 (Selection Logic, L=1)
        X'01'                   1 (OR)

        X'1E'                   PD Opening Tag 1 (List Of Selection Criteria)
                X'09'               SD Context Tag 0 (Property Identifier, L=1)
                X'4D'               77 (OBJECT_NAME)
                X'29'               SD Context Tag 2 (Relation Specifier)
                X'00'               0 (EQUAL)
                X'3E'               PD Opening Tag 3 (Comparison Value)
                        X'7D'           Application Tag 7 (Character String, L>4)
                        X'0C'           Extended Length=12
                        X'00'           ANSI X3.4 Encoding
                        X'432A205072657373757265'  "C* Pressure"
                X'3F'               PD Closing Tag 3 (Comparison Value)

                X'09'               SD Context Tag 0 (Property Identifier, L=1)
                X'4D'               77 (OBJECT_NAME)
                X'29'               SD Context Tag 2 (Relation Specifier)
                X'00'               0 (EQUAL)
                X'3E'               PD Opening Tag 3 (Comparison Value)
                        X'7D'           Application Tag 7 (Character String, L>4)
                        X'10'           Extended Length=16
                        X'00'           ANSI X3.4 Encoding
                        X'41433F20537570706C792054656D70'  "AC? Supply Temp"
                X'3F'               PD Closing Tag 3 (Comparison Value)
        X'1F'                   PD Closing Tag 1 (List Of Selection Criteria)
X'0F'                       PD Closing Tag 0 (Object Selection Criteria)
```

```
X'1E'                   PD Opening Tag 1 (List Of Property References)
      X'09'                 SD Context Tag 0 (Property Identifier, L=1)
      X'4D'                 77 (OBJECT_NAME)
X'1F'                   PD Closing Tag 1 (List Of Property References)
```

Assuming this service procedure executes correctly, a complex acknowledgement is returned:

```
X'30'                   PDU Type=3 (BACnet-ComplexACK-PDU, SEG=0, MOR=0)
X'54'                   Invoke ID=84
X'0D'                   Service Ack Choice=13 (ReadPropertyConditional-ACK)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'00000004'             Analog Input, Instance Number=4
X'1E'                   PD Opening Tag 1 (List Of Results)
      X'29'                 SD Context Tag 2 (Property Identifier, L=1)
      X'4D'                 77 (OBJECT_NAME)
      X'4E'                 PD Opening Tag 4 (Property Value)
            X'75'               Application Tag 7 (Character String, L>4)
            X'10'               Extended Length=16
            X'00'               ANSI X3.4 Encoding
            X'41433120537570706C792054656D70'  "AC1 Supply Temp"
      X'4F'                 PD Closing Tag 4 (Property Value)
X'1F'                   PD Closing Tag 1 (List Of Results)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'00000007'             Analog Input, Instance Number=7
X'1E'                   PD Opening Tag 1 (List Of Results)
      X'29'                 SD Context Tag 2 (Property Identifier, L=1)
      X'4D'                 77 (OBJECT_NAME)
      X'4E'                 PD Opening Tag 4 (Property Value)
            X'75'               Application Tag 7 (Character String, L>4)
            X'0E'               Extended Length=14
            X'00'               ANSI X3.4 Encoding
            X'435750312050726573737375726'  "CWP1 Pressure"
      X'4F'                 PD Closing Tag 4 (Property Value)
X'1F'                   PD Closing Tag 1 (List Of Results)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'00000008'             Analog Input, Instance Number=8
X'1E'                   PD Opening Tag 1 (List Of Results)
      X'29'                 SD Context Tag 2 (Property Identifier, L=1)
      X'4D'                 77 (OBJECT_NAME)
      X'4E'                 PD Opening Tag 4 (Property Value)
            X'75'               Application Tag 7 (Character String, L>4)
            X'19'               Extended Length=25
            X'00'               ANSI X3.4 Encoding
            X'4368696C6C6572203120467265F6E2050726573737375726'  "Chiller 1 Freon Pressure"
      X'4F'                 PD Closing Tag 4 (Property Value)
X'1F'                   PD Closing Tag 1 (List Of Results)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'0000000A'             Analog Input, Instance Number=10
```

```
X'1E'                   PD Opening Tag 1 (List Of Results)
    X'29'                   SD Context Tag 2 (Property Identifier, L=1)
    X'4D'                   77 (OBJECT_NAME)
    X'4E'                   PD Opening Tag 4 (Property Value)
        X'75'                   Application Tag 7 (Character String, L>4)
        X'10'                   Extended Length=16
        X'00'                   ANSI X3.4 Encoding
        X'41433220537570706C792054656D70'  "AC2 Supply Temp"
    X'4F'                   PD Closing Tag 4 (Property Value)
X'1F'                   PD Closing Tag 1 (List Of Results)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'0000000C'             Analog Input, Instance Number=12
X'1E'                   PD Opening Tag 1 (List Of Results)
    X'29'                   SD Context Tag 2 (Property Identifier, L=1)
    X'4D'                   77 (OBJECT_NAME)
    X'4E'                   PD Opening Tag 4 (Property Value)
        X'75'                   Application Tag 7 (Character String, L>4)
        X'10'                   Extended Length=16
        X'00'                   ANSI X3.4 Encoding
        X'41433320537570706C792054656D70'  "AC3 Supply Temp"
    X'4F'                   PD Closing Tag 4 (Property Value)
X'1F'                   PD Closing Tag 1 (List Of Results)
```

F.3.7 Encoding for Example E.3.7 - ReadPropertyMultiple Service

Example 1: Reading multiple properties of a single object.

```
X'00'                   PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X'04'                   Maximum APDU Size Accepted=1024 octets
X'F1'                   Invoke ID=241
X'0E'                   Service Choice=14 (ReadPropertyMultiple-Request)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'00000010'             Analog Input, Instance Number=16
X'1E'                   PD Opening Tag 1 (List Of Property References)
    X'09'                   SD Context Tag 0 (Property Identifier, L=1)
    X'55'                   85 (PRESENT_VALUE)

    X'09'                   SD Context Tag 0 (Property Identifier, L=1)
    X'67'                   103 (RELIABILITY)
X'1F'                   PD Closing Tag 1 (List Of Property References)
```

Assuming this service procedure executes correctly, a complex acknowledgement is returned:

```
X'30'                   PDU Type=3 (BACnet-ComplexACK-PDU, SEG=0, MOR=0)
X'F1'                   Invoke ID=241
X'0E'                   Service ACK Choice=14 (ReadPropertyMultiple-ACK)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'00000010'             Analog Input, Instance Number=16
X'1E'                   PD Opening Tag 1 (List Of Results)
    X'29'                   SD Context Tag 2 (Property Identifier, L=1)
```

```
        X'55'                   85 (PRESENT_VALUE)
        X'4E'                   PD Opening Tag 4 (Property Value)
                X'44'               Application Tag 4 (Real, L=4)
                X'4290999A'         72.3
        X'4F'                   PD Closing Tag 4 (Property Value)

        X'29'                   SD Context Tag 2 (Property Identifier, L=1)
        X'67'                   103 (RELIABILITY)
        X'4E'                   PD Opening Tag 4 (Property Value)
                X'91'               Application Tag 9 (Enumerated, L=1)
                X'00'               0 (NO_FAULT_DETECTED)
        X'4F'                   PD Closing Tag 4 (Property Value)
X'1F'                   PD Closing Tag 1 (List Of Results)
```

Example 2: Reading properties of several objects.

```
X'00'                   PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0,  SA=0)
X'04'                   Maximum APDU Size Accepted=1024 octets
X'02'                   Invoke ID=2
X'0E'                   Service Choice=14 (ReadPropertyMultiple-Request)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'00000021'             Analog Input, Instance Number=33
X'1E'                   PD Opening Tag 1 (List Of Property References)
        X'09'               SD Context Tag 0 (Property Identifier, L=1)
        X'55'               85 (PRESENT_VALUE)
X'1F'                   PD Closing Tag 1 (List Of Property References)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'00000032'             Analog Input, Instance Number=50
X'1E'                   PD Opening Tag 1 (List Of Property References)
        X'09'               SD Context Tag 0 (Property Identifier, L=1)
        X'55'               85 (PRESENT_VALUE)
X'1F'                   PD Closing Tag 1 (List Of Property References)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'00000023'             Analog Input, Instance Number=35
X'1E'                   PD Opening Tag 1 (List Of Property References)
        X'09'               SD Context Tag 0 (Property Identifier, L=1)
        X'55'               85 (PRESENT_VALUE)
X'1F'                   PD Closing Tag 1 (List Of Property References)
```

Assuming this service procedure executes correctly, a complex acknowledgement is returned:

```
X'30'                   PDU Type=3 (BACnet-ComplexACK-PDU, SEG=0, MOR=0)
X'02'                   Invoke ID=2
X'0E'                   Service ACK Choice=14 (ReadPropertyMultiple-ACK)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'00000021'             Analog Input, Instance Number=33
X'1E'                   PD Opening Tag 1 (List Of Results)
        X'29'               SD Context Tag 2 (Property Identifier, L=1)
        X'55'               85 (PRESENT_VALUE)
```

```
        X'4E'                   PD Opening Tag 4 (Property Value)
              X'44'                 Application Tag 4 (Real, L=4)
              X'42293333'           42.3
        X'4F'                   PD Closing Tag 4 (Property Value)
X'1F'                   PD Closing Tag 1 (List Of Results)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'00000032'             Analog Input, Instance Number=50
X'1E'                   PD Opening Tag 1 (List Of Results)
        X'29'               SD Context Tag 2 (Property Identifier, L=1)
        X'55'               85 (PRESENT_VALUE)
        X'5E'               PD Opening Tag 5 (Property Access Error)
              X'91'             Application Tag 9 (Enumerated, L=1) (Error Class)
              X'01'             1 (OBJECT)
              X'91'             Application Tag 9 (Enumerated, L=1) (Error Code)
              X'1F'             31 (UNKNOWN_OBJECT)
        X'5F'               PD Closing Tag 5 (Property Access Error)
X'1F'                   PD Closing Tag 1 (List Of Results)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'00000023'             Analog Input, Instance Number=35
X'1E'                   PD Opening Tag 1 (List Of Results)
        X'29'               SD Context Tag 2 (Property Identifier, L=1)
        X'55'               85 (PRESENT_VALUE)
        X'4E'               PD Opening Tag 4 (Property Value)
              X'44'             Application Tag 4 (Real, L=4)
              X'43D9D99A'       435.7
        X'4F'               PD Closing Tag 4 (Property Value)
X'1F'                   PD Closing Tag 1 (List Of Results)
```

F.3.8 Encoding for Example E.3.8 - WriteProperty Service

```
X'00'                   PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X'04'                   Maximum APDU Size Accepted=1024 octets
X'59'                   Invoke ID=89
X'0F'                   Service Choice=15 (WriteProperty-Request)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'00800001'             Analog Value, Instance Number=1
X'19'                   SD Context Tag 1 (Property Identifier, L=1)
X'55'                   85 (PRESENT_VALUE)
X'3E'                   PD Opening Tag 3 (Property Value)
        X'44'               Application Tag 4 (Real, L=4)
        X'43340000'         Property Value=180.0
X'3F'                   PD Closing Tag 3 (Property Value)
```

Assuming the service procedure executes correctly, a simple acknowledgement is returned:

```
X'20'                   PDU Type=2 (BACnet-SimpleACK-PDU)
X'59'                   Invoke ID=89
X'0F'                   Service ACK Choice=15 (WriteProperty)
```

F.3.9 Encoding for Example E.3.9 - WritePropertyMultiple Service

```
X'00'                   PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X'04'                   Maximum APDU Size Accepted=1024 octets
X'01'                   Invoke ID=1
X'10'                   Service Choice=16 (WritePropertyMultiple-Request)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'00800005'             Analog Value, Instance Number=5
X'1E'                   PD Opening Tag 1 (List Of Properties)
    X'09'                   SD Context Tag 0 (Property Identifier, L=1)
    X'55'                   85 (PRESENT_VALUE)
    X'2E'                   PD Opening Tag 2 (Property Value)
        X'44'                   Application Tag 4 (Real, L=4)
        X'42860000'             67.0
    X'2F'                   PD Closing Tag 2 ( Property Value)
X'1F'                   PD Closing Tag 1 (List Of Properties)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'00800006'             Analog Value, Instance Number=6
X'1E'                   PD Opening Tag 1 (List Of Properties)
    X'09'                   SD Context Tag 0 (Property Identifier, L=1)
    X'55'                   85 (PRESENT_VALUE)
    X'2E'                   PD Opening Tag 2 (Property Value)
        X'44'                   Application Tag 4 (Real, L=4)
        X'42860000'             67.0
    X'2F'                   PD Closing Tag 2 (Property Value)
X'1F'                   PD Closing Tag 1 (List Of Properties)

X'0C'                   SD Context Tag 0 (Object Identifier, L=4)
X'00800007'             Analog Value, Instance Number=7
X'1E'                   PD Opening Tag 1 (List Of Properties)
    X'09'                   SD Context Tag 0 (Property Identifier, L=1)
    X'55'                   85 (PRESENT_VALUE)
    X'2E'                   PD Opening Tag 2 (Property Value)
        X'44'                   Application Tag 4 (Real, L=4)
        X'42900000'             72.0
    X'2F'                   PD Closing Tag 2 (Property Value)
X'1F'                   PD Closing Tag 1 (List Of Properties)
```

Assuming the service procedure executes correctly, a simple acknowledgement is returned:

```
X'20'                   PDU Type=2 (BACnet-SimpleACK-PDU)
X'01'                   Invoke ID=1
X'10'                   Service ACK Choice=16 (WritePropertyMultiple)
```

F.4 Example Encodings for Remote Device Management Services

F.4.1 Encoding for Example E.4.1 - DeviceCommunicationControl Service

```
X'00'                   PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X'04'                   Maximum APDU Size Accepted=1024 octets
X'05'                   Invoke ID=5
X'11'                   Service Choice=17 (DeviceCommunicationControl-Request)
```

X'09'	SD Context Tag 0 (Time Duration, L=1)
X'05'	5
X'19'	SD Context Tag 1 (Enable-Disable, L=1)
X'01'	1 (DISABLE)
X'2D'	SD Context Tag 2 (Password, L>4)
X'08'	Extended Length=8
X'00'	ANSI X3.4 Encoding
X'23656762646621'	"#egbdf!"

Assuming the service procedure executes correctly, a simple acknowledgement is returned:

X'20'	PDU Type=2 (BACnet-SimpleACK-PDU)
X'05'	Invoke ID=5
X'11'	Service ACK Choice=17 (DeviceCommunicationControl)

F.4.2 Encoding for Example E.4.2 - ConfirmedPrivateTransfer Service

X'00'		PDU Type=0 (BACnet Confirmed-Request-PDU) SEG=0, MOR=0, SA=0)
X'04'		Maximum APDU Size Accepted=1024 octets
X'55'		Invode ID=85
X'12'		Service Choice=18 (ConfirmedPrivateTransfer)
X'09'		SD Context Tag 0 (Vendor ID, L=1)
X'19'		25 (XYZ Controls Company Limited)
X'19'		SD Context Tag 1 (Service Number, L=1)
X'08'		8 (XYZ Proprietary Service #8)
X'2E'		PD Opening Tag 2 (Service Parameters)
	X'44'	Application Tag 4 (Real, L=4)
	X'4290CCCD'	72.4
	X'62'	Application Tag 6 (Octet String, L=2)
	X'1649'	X'1649'
X'2F'		PD Closing Tag 2 (Service Parameters)

Assuming that the service is successfully executed but no results need to be returned to the requesting BACnet-user, a 'Result(+)' primitive will be returned using a BACnet-ComplexACK-PDU:

X'30'	PDU Type=3 (BACnet-ComplexACK-PDU, SEG=0, MOR=0)
X'55'	Invoke ID=85
X'12'	Service Choice=18 (ConfirmedPrivateTransfer-ACK)
X'09'	SD Context Tag 0 (Vendor ID, L=1)
X'19'	25
X'19'	SD Context Tag 1 (Service Number, L=1)
X'08'	8

F.4.3 Encoding for Example E.4.3 - UnconfirmedPrivateTransfer Service

X'10'	PDU Type=1 (BACnet-Unconfirmed-Request-PDU)
X'04'	Service Choice=4 (UnconfirmedPrivateTransfer-Request)
X'09'	SD Context Tag 0 (Vendor ID, L=1)
X'19'	25
X'19'	SD Context Tag 1 (Service Number, L=1)
X'08'	8
X'2E'	PD Opening Tag 2 (Service Parameters)

```
        X'44'               Application Tag 4 (Real, L=4)
        X'4290CCCD'         72.4
        X'62'               Application Tag 6 (Octet String, L=2)
        X'1649'             X'1649'
X'2F'               PD Closing Tag 2 (Service Parameters)
```

F.4.4 Encoding for Example E.4.4 - ReinitializeDevice Service

```
X'00'               PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X'01'               Maximum APDU Size Accepted=128 octets
X'02'               Invoke ID=2
X'14'               Service Choice=20 (ReinitializeDevice-Request)

X'09'               SD Context Tag 0 (Reinitialized State Of Device, L=1)
X'01'               1 (WARMSTART)
X'1D'               SD Context Tag 1 (Password, L>4)
X'09'               Extended Length=9
X'00'               ANSI X3.4 Encoding
X'4162436445664768' "AbCdEfGh"
```

Assuming this service procedure executes correctly, a simple acknowledgement is returned:

```
X'20'               PDU Type=2 (BACnet-SimpleACK-PDU)
X'02'               Invoke ID=2
X'14'               Service ACK Choice=20 (ReinitializeDevice)
```

F.4.5 Encoding for Example E.4.5 - ConfirmedTextMessageService

```
X'00'               PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X'01'               Maximum APDU Size Accepted=128 octets
X'03'               Invoke ID=3
X'13'               Service Choice=19 (ConfirmedTextMessage-Request)

X'0C'               SD Context Tag 0 (Text Message Source Device, L=4)
X'02000005'         Device, Instance Number=5
X'29'               SD Context Tag 2 (Message Priority, L=1)
X'00'               0 (NORMAL)
X'3D'               SD Context Tag 3 (Message, L>4)
        X'18'               Extended Length=24
        X'00'               ANSI X3.4 Encoding
        X'504D20726571756972656420666F722050554D50333437' "PM required for PUMP347"
```

Assuming the service procedure executes correctly, a simple acknowledgement is returned:

```
X'20'               PDU Type=2  (BACnetSimpleACK-PDU)
X'03'               Invoke ID=3
X'13'               Service ACK Choice=19 (ConfirmedTextMessage)
```

F.4.6 Encoding for Example E.4.6 - UnconfirmedTextMessage Service

```
X'10'               PDU Type=1 (BACnet-Unconfirmed-Request-PDU)
X'05'               Service Choice=5 (UnconfirmedTextMessage-Request)

X'0C'               SD Context Tag 0 (Text Message Source Device, L=4)
X'02000005'         Device, Instance Number=5
```

```
X'29'                   SD Context Tag 2 (Message Priority, L=1)
 X'00'                  0 (NORMAL)
X'3D'                   SD Context Tag 3 (Message, L>4)
        X'18'                   Extended Length=24
        X'00'                   ANSI X3.4 Encoding
        X'504D207265717569726564206 66F722050554D50333437' "PM required for PUMP347"
```

Note that no response is required for this message since it is of type unconfirmed.

F.4.7 Encoding for Example E.4.7 - TimeSynchronization Service

```
X'10'                   PDU Type=1 (BACnet-Unconfirmed-Service-Request-PDU)
X'06'                   Service Choice=6 (TimeSynchronization-Request)

X'A4'                   Application Tag 10 (Date, L=4)
X'5C0B11FF'             November 17, 1992 (Day of Week Unspecified)
X'B4'                   Application Tag 11 (Time, L=4)
X'162D1E46'             22:45:30.70
```

F.4.8 Encoding for Example E.4.8 - Who-Has and I-Have Services

Example 1: Locating the device that contains an object for which the Object_Name is known.

```
X'10'                   PDU Type=1 (Unconfirmed-Service-Request-PDU)
X'07'                   Service Choice=7 (Who-Has-Request)

X'3D'                   SD  Context Tag 3 (Object Name, L>4)
        X'07'                   Extended Length=7
        X'00'                   ANSI X3.4 Encoding
        X'4F4154656D70'         "OATemp"
```

Assuming that exactly one other device has such an object, the following I-Have service indication would be received.

```
X'10'                   PDU Type=1 (Unconfirmed-Service-Request-PDU)
X'01'                   Service Choice=1 (I-Have-Request)

X'C4'                   Application Tag 12 (Object Identifier, L=4) (Device Identifier)
X'02000008'             Device, Instance Number=8
X'C4'                   Application Tag 12 (Object Identifier, L=4) (Object Identifier)
X'00000003'             Analog Input, Instance Number=3
X'75'                   Application Tag 7 (Character String, L>4) (Object Name)
        X'07'                   Extended Length=7
        X'00'                   ANSI X3.4 Encoding
        X'4F4154656D70'         "OATemp"
```

Example 2: Locating the device that contains an object for which the Object_Identifier is known.

```
X'10'                   PDU Type=1 (Unconfirmed-Service-Request-PDU)
X'07'                   Service Choice=7 (Who-Has-Request)

X'2C'                   SD Context Tag 2 (Object Identifier, L=4)
X'00000003'             Analog Input, Instance Number=3
```

Assuming that exactly one other device has such an object, the following I-Have service indication would be received.

X'10'	PDU Type=1 (Unconfirmed-Service-Request-PDU)
X'01'	Service Choice=1 (I-Have-Request)
X'C4'	Application Tag 12 (Object Identifier, L=4) (Device Identifier)
X'02000008'	Device, Instance Number=8
X'C4'	Application Tag 12 (Object Identifier, L=4) (Object Identifier)
X'00000003'	Analog Input, Instance Number=3
X'75'	Application Tag 7 (Character String, L>4) (Object Name)
X'07'	Extended Length=7
X'00'	ANSI X3.4 Encoding
X'4F4154656D70'	"OATemp"

F.4.9 Encoding for Example E.4.9 - Who-Is and I-Am Services

Example 1: Establishing the network address of a device with a known Device object identifier, i.e., instance number.

X'10'	PDU Type=1 (Unconfirmed-Service-Request-PDU)
X'08'	Service Choice=8 (Who-Is-Request)
X'09'	SD Context Tag 0 (Device Instance Range Low Limit, L=1)
X'03'	3
X'19'	SD Context Tag 1 (Device Instance Range High Limit, L=1)
X'03'	3

Assuming that there is such a device on the network, it responds some time later using the I-Am service:

X'10'	PDU Type=1 (Unconfirmed-Service-Request-PDU)
X'00'	Service Choice=0 (I-Am-Request)
X'C4'	Application Tag 12 (Object Identifier, L=4) (I-Am Device Identifier)
X'02000003'	Device, Instance Number=3
X'22'	Application Tag 2 (Unsigned Integer, L=2) (Max APDU Length Accepted)
X'0400'	1024
X'91'	Application Tag 9 (Enumerated, L=1) (Segmentation Supported)
X'03'	3 (NO_SEGMENTATION)
X'21'	Application Tag 2 (Unsigned Integer, L=1) (Vendor ID)
X'63'	99

Example 2: Finding out about all network devices.

X'10'	PDU Type=1 (Unconfirmed-Service-Request-PDU)
X'08'	Service Choice=8 (Who-Is-Request)

Each device on the network responds using the I-Am service:

X'10'	PDU Type=1 (Unconfirmed-Service-Request-PDU)
X'00'	Service Choice=0 (I-Am-Request)
X'C4'	Application Tag 12 (Object Identifier, L=4) (I-Am Device Identifier)
X'02000001'	Device, Instance Number=1
X'22'	Application Tag 2 (Unsigned Integer, L=2) (Max APDU Length Accepted)
X'01E0'	480
X'91'	Application Tag 9 (Enumerated, L=1) (Segmentation Supported)
X'01'	1 (SEGMENTED_TRANSMIT)

X'21'	Application Tag 2 (Unsigned Integer, L=1) (Vendor ID)
X'63'	99
X'10'	PDU Type=1 (Unconfirmed-Service-Request-PDU)
X'00'	Service Choice=0 (I-Am-Request)
X'C4'	Application Tag 12 (Object Identifier, L=4) (I-Am Device Identifier)
X'02000002'	Device, Instance Number=2
X'21'	Application Tag 2 (Unsigned Integer, L=1) (Max APDU Length Accepted)
X'CE'	206
X'91'	Application Tag 9 (Enumerated, L=1) (Segmentation Supported)
X'02'	2 (SEGMENTED_RECEIVE)
X'21'	Application Tag 2 (Unsigned Integer, L=1) (Vendor ID)
X'21'	33
X'10'	PDU Type=1 (Unconfirmed-Service-Request-PDU)
X'00'	Service Choice=0 (I-Am-Request)
X'C4'	Application Tag 12 (Object Identifier, L=4) (I-Am Device Identifier)
X'02000003'	Device, Instance Number=3
X'22'	Application Tag 2 (Unsigned Integer, L=2) (Max APDU Length Accepted)
X'0400'	1024
X'91'	Application Tag 9 (Enumerated, L=1) (Segmentation Supported)
X'03'	3 (NO_SEGMENTATION)
X'21'	Application Tag 2 (Unsigned Integer, L=1) (Vendor ID)
X'63'	99
X'10'	PDU Type=1 (Unconfirmed-Service-Request-PDU)
X'00'	Service Choice=0 (I-Am-Request)
X'C4'	Application Tag 12 (Object Identifier, L=4) (I-Am Device Identifier)
X'02000004'	Device, Instance Number=4
X'21'	Application Tag 2 (Unsigned Integer, L=1) (Max APDU Length Accepted)
X'80'	128
X'91'	Application Tag 9 (Enumerated, L=1) (Segmentation Supported)
X'00'	0 (SEGMENTED_BOTH)
X'21'	Application Tag 2 (Unsigned Integer, L=1) (Vendor ID)
X'42'	66

F.5 Encoding for Example E.5 - Virtual Terminal Services

Establishing a VT session:

X'00'	PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X'01'	Maximum APDU Size Accepted=128 octets
X'50'	Invoke ID=80
X'15'	Service Choice=21 (VT-Open-Request)
X'91'	Application Tag 9 (Enumerated, L=1) (VT Class)
X'01'	1 (ANSI_X3.64)
X'21'	Application Tag 2 (Unsigned Integer, L=1) (Local VT Session Identifier)
X'05'	5

Assuming that the target device can create a new VT-session, a complex acknowledgement is returned:

X'30'	PDU Type=3 (BACnet-ComplexACK-PDU, SEG=0, MOR=0)
X'50'	Invoke ID=80
X'15'	Service Choice=21 (VT-Open-ACK)
X'21'	Application Tag 2 (Unsigned Integer, L=1) (Remote VT Session Identifier)
X'1D'	29

Terminal sign-on. The target device sends a prompt:

X'00'	PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X'01'	Maximum APDU Size Accepted=128 octets
X'51'	Invoke ID=81
X'17'	Service Choice=23 (VT-Data-Request)
X'21'	Application Tag 2 (Unsigned Integer, L=1) (VT Session Identifier)
X'05'	5
X'65'	Application Tag 6 (Octet String, L>4) (VT New Data)
X'12'	Extended Length=18
X'0D0A456E7465722055736572204E616D653A'	"{cr}{lf}Enter User Name:"
X'21'	Application Tag 2 (Unsigned Integer, L=1) (VT Data Flag)
X'00'	0

Assuming that the operator interface device receives the data correctly, a complex acknowledgement is returned:

X'30'	PDU Type=3 (BACnet-ComplexACK-PDU, SEG=0, MOR=0)
X'51'	Invoke ID=81
X'17'	Service Choice=23 (VT-Data-ACK)
X'09'	SD Context Tag 0 (All New Data Accepted, L=1)
X'01'	1 (TRUE)

Entering user name:

X'00'	PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X'01'	Maximum APDU Size Accepted=128 octets
X'52'	Invoke ID=82
X'17'	Service Choice=23 (VT-Data-Request)
X'21'	Application Tag 2 (Unsigned Integer, L=1) (VT Session Identifier)
X'1D'	29
X'65'	Application Tag 6 (Octet String, L>4) (VT New Data)
X'05'	Extended Length=5
X'465245440D'	"FRED{cr}"
X'21'	Application Tag 2 (Unsigned Integer, L=1) (VT Data Flag)
X'00'	0

To which the target device would respond:

X'30'	PDU Type=3 (BACnet-ComplexACK-PDU, SEG=0, MOR=0)
X'52'	Invoke ID=82
X'17'	Service Choice=23 (VT-Data-ACK)
X'09'	SD Context Tag 0 (All New Data Accepted, L=1)
X'01'	1 (TRUE)

Entering password:

X'00'	PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X'01'	Maximum APDU Size Accepted=128 octets
X'53'	Invoke ID=83
X'17'	Service Choice=23 (VT-Data-Request)
X'21'	Application Tag 2 (Unsigned Integer, L=1) (VT Session Identifier)
X'05'	5
X'65'	Application Tag 6 (Octet String, L>4) (VT New Data)
X'15'	Extended Length=21
X'465245440D0A456E7465722050617373776F72643A'	"FRED{cr}{lf}Enter Password:"
X'21'	Application Tag 2 (Unsigned Integer, L=1) (VT Data Flag)
X'01'	1

To which the target device would respond:

X'30'	PDU Type=3 (BACnet-ComplexACK-PDU, SEG=0, MOR=0)
X'53'	Invoke ID=83
X'17'	Service Choice=23 (VT-Data-ACK)
X'09'	SD Context Tag 0 (All New Data Accepted, L=1)
X'01'	1 (TRUE)

Terminal sign-off:

X'00'	PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X'01'	Maximum APDU Size Accepted=128 octets
X'54'	Invoke ID=84
X'16'	Service Choice=22 (VT-Close-Request)
X'21'	Application Tag 2 (Unsigned Integer, L=1) (List Of Remote VT Session Identifiers)
X'1D'	29

Response:

X'20'	PDU Type=2 (BACnet-SimpleACK-PDU)
X'54'	Invoke ID=84
X'16'	Service ACK Choice=22 (VT-Close)

F.6 Encoding for Example E.6 - Security Services

Device A sends a Request Key service request to the key server:

X'00'	PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X'05'	Maximum APDU Size Accepted=1470 octets
X'0F'	Invoke ID=15
X'19'	Service Choice=25 (RequestKey-Request)

--- Note that all octets from here to the end of the APDU are to be enciphered ---

X'C4'	*Application Tag 12 (Object Identifier, L=4) (Requesting Device Identifier)*
X'02000001'	*Device, Instance Number=1*
X'22'	*Application Tag 2 (Unsigned Integer, L=2) (Network Number)*
X'0002'	*2*
X'61'	*Application Tag 6 (Octet String, L=1) (MAC Address)*

X'11'	*17*
X'C4'	*Application Tag 12 (Object Identifier, L=4) (Remote Device Identifier)*
X'02000002'	*Device, Instance Number=2*
X'22'	*Application Tag 2 (Unsigned Integer, L=2) (Network Number)*
X'0002'	*2*
X'61'	*Application Tag 6 (Octet String, L=1) (MAC Address)*
X'22'	*34*

The key server then seeks to authenticate the Request Key request:

X'00'	PDU Type=0 (BACnet-Confirmed-Request-PDU, SEG=0, MOR=0, SA=0)
X'05'	Maximum APDU Size Accepted=1470 octets
X'01'	Invoke ID=1
X'18'	Service Choice=24 (Authenticate-Request)

— Note that all octets from here to the end of the APDU are to be enciphered —

X'0C'	*SD Context Tag 0 (Pseudo Random Number, L=4)*
X'12345678'	*305,419,896*
X'19'	*SD Context Tag 1 (Expected Invoke ID, L=1)*
X'0F'	*15*

The originator of the Request Key service request responds to the authenticate request with a complex acknowledgement:

X'30'	PDU Type=3 (BACnet-ComplexACK-PDU, SEG=0, MOR=0)
X'01'	Invoke ID=01
X'18'	Service Choice=24 (Authenticate-ACK)

— Note that all octets from here to the end of the APDU are to be enciphered —

X'24'	*Application Tag 2 (Unsigned Integer, L=4) (Modified Random Number)*
X'93B5D7F1'	*2,478,168,049*

ANNEX G - CALCULATION OF CRC (INFORMATIVE)

(This annex is not part of this standard but is included for informative purposes only.)

Historically, CRC generators have been implemented as shift registers with exclusive OR feedback. This provides an inexpensive way to process CRC information on a serial bit stream in hardware. Since commercial UARTs do not provide hardware calculation of CRC, UART-based protocols such as those described in Clauses 9 and 10 must perform this calculation with software. While this can be done one bit at a time, simulating a shift register with feedback, a much more efficient algorithm is possible that computes the CRC on an entire octet at once. This annex shows how the CRC may be computed in this manner. This algorithm is presented as an example and is not intended to restrict the vendor's implementation of the CRC calculation.

G.1 Calculation of the Header CRC

We begin with the diagram of a hardware CRC generator as shown in Figure G-1. The polynomial used is

$$X^8 + X^7 + 1$$

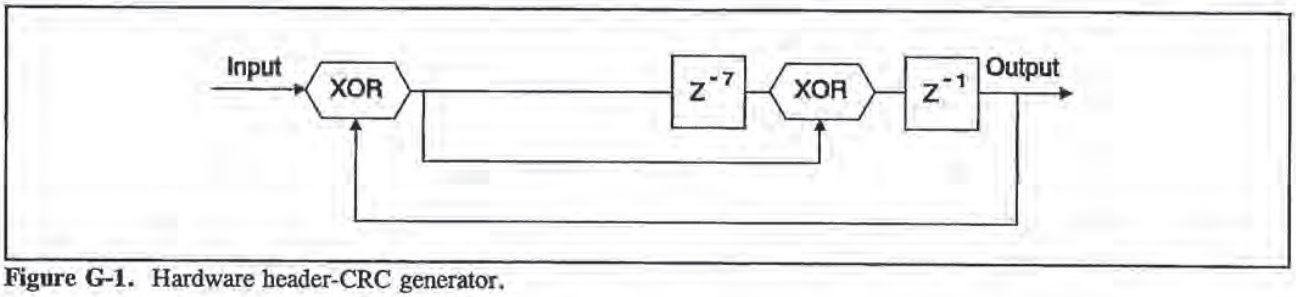

Figure G-1. Hardware header-CRC generator.

The hardware implementation operates on a serial bit stream, whereas our calculation must operate on entire octets. To this end, we follow the operation of the circuit through eight bits of data. The CRC shift register is initialized to X0 (input end) to X7 (output end), the input data is D0 to D7 (least significant bit first, as transmitted and received by a UART). Within each block below, the terms are exclusive OR'ed vertically.

input	register contents							
D0	X0	X1	X2	X3	X4	X5	X6	X7
D1		X0	X1	X2	X3	X4	X5	X6
	D0							D0
	X7							X7
D2			X0	X1	X2	X3	X4	X5
	D1	D0						D1
	X6	X7						X6
	D0							D0
	X7							X7
	(continued on following page)							
D3				X0	X1	X2	X3	X4
	D2	D1	D0					D2
	X5	X6	X7					X5
	D1	D0						D1
	X6	X7						X6
	D0							D0
	X7							X7
D4					X0	X1	X2	X3

```
|  D3  D2  D1  D0                  D3  |
|  X4  X5  X6  X7                  X4  |
|  D2  D1  D0                      D2  |
|  X5  X6  X7                      X5  |
|  D1  D0                          D1  |
|  X6  X7                          X6  |
|  D0                              D0  |
|  X7                              X7  |
|-------------------------------------|
D5|                        X0  X1   X2  |
|  D4  D3  D2  D1  D0              D4  |
|  X3  X4  X5  X6  X7              X3  |
|  D3  D2  D1  D0                  D3  |
|  X4  X5  X6  X7                  X4  |
|  D2  D1  D0                      D2  |
|  X5  X6  X7                      X5  |
|  D1  D0                          D1  |
|  X6  X7                          X6  |
|  D0                              D0  |
|  X7                              X7  |
|-------------------------------------|
D6|                            X0   X1  |
|  D5  D4  D3  D2  D1  D0          D5  |
|  X2  X3  X4  X5  X6  X7          X2  |
|  D4  D3  D2  D1  D0              D4  |
|  X3  X4  X5  X6  X7              X3  |
|  D3  D2  D1  D0                  D3  |
|  X4  X5  X6  X7                  X4  |
|  D2  D1  D0                      D2  |
|  X5  X6  X7                      X5  |
|  D1  D0                          D1  |
|  X6  X7                          X6  |
|  D0                              D0  |
|  X7                              X7  |
|-------------------------------------|
```

(continued on following page)

D7								X0
	D6	D5	D4	D3	D2	D1	D0	D6
	X1	X2	X3	X4	X5	X6	X7	X1
	D5	D4	D3	D2	D1	D0		D5
	X2	X3	X4	X5	X6	X7		X2
	D4	D3	D2	D1	D0			D4
	X3	X4	X5	X6	X7			X3
	D3	D2	D1	D0				D3
	X4	X5	X6	X7				X4
	D2	D1	D0					D2
	X5	X6	X7					X5
	D1	D0						D1
	X6	X7						X6
	D0							D0
	X7							X7
								D0
								X7
	D7	D6	D5	D4	D3	D2	D1	D7
	X0	X1	X2	X3	X4	X5	X6	X0
	D6	D5	D4	D3	D2	D1	D0	D6
	X1	X2	X3	X4	X5	X6	X7	X1
	D5	D4	D3	D2	D1	D0		D5
	X2	X3	X4	X5	X6	X7		X2
	D4	D3	D2	D1	D0			D4
	X3	X4	X5	X6	X7			X3
	D3	D2	D1	D0				D3
	X4	X5	X6	X7				X4
	D2	D1	D0					D2
	X5	X6	X7					X5
	D1	D0						D1
	X6	X7						X6
	D0							D0
	X7							X7

The final block above is the net result of shifting an octet of data through the CRC generator. By performing the indicated exclusive OR operations, an octet can be processed into the CRC. In order to simplify the required shifting operations, we define X0 to be the most significant bit of the parallel representation and X7 to be the least significant. Since most microprocessors number bits in the opposite order, we rename the bits to correspond to standard microprocessor bit nomenclature. Let C7 = X0, C6 = X1, etc. The final block in the previous table thus becomes:

X0	X1	X2	X3	X4	X5	X6	X7
C7	C6	C5	C4	C3	C2	C1	C0
							D0
							C0
D7	D6	D5	D4	D3	D2	D1	D7
C7	C6	C5	C4	C3	C2	C1	C7
D6	D5	D4	D3	D2	D1	D0	D6
C6	C5	C4	C3	C2	C1	C0	C6
D5	D4	D3	D2	D1	D0		D5
C5	C4	C3	C2	C1	C0		C5
D4	D3	D2	D1	D0			D4
C4	C3	C2	C1	C0			C4
D3	D2	D1	D0				D3
C3	C2	C1	C0				C3
D2	D1	D0					D2
C2	C1	C0					C2
D1	D0						D1
C1	C0						C1
D0							D0
C0							C0

or, rearranging vertically (since exclusive OR is commutative) to minimize computation, and recognizing that any term exclusive OR'ed with itself may be eliminated:

D7	D6	D5	D4	D3	D2	D1	
C7	C6	C5	C4	C3	C2	C1	
D6	D5	D4	D3	D2	D1	D0	D7
C6	C5	C4	C3	C2	C1	C0	C7
D5	D4	D3	D2	D1	D0		D6
C5	C4	C3	C2	C1	C0		C6
D4	D3	D2	D1	D0			D5
C4	C3	C2	C1	C0			C5
D3	D2	D1	D0				D4
C3	C2	C1	C0				C4
D2	D1	D0					D3
C2	C1	C0					C3
D1	D0						D2
C1	C0						C2
D0							D1
C0							C1

In operation, the CRC register C7-C0 is initialized to all ones. During transmission, the Frame Type, Destination Address, Source Address, and Length are passed though the calculation before transmission. The ones complement of the final CRC register value is then transmitted. At the receiving end, the Frame Type, Destination Address, Source Address, Length octets, and the received CRC octet are passed though the calculation. If all octets are received correctly, then the final value of the receiver's CRC register will be the constant X'55'.

As an example of usage, consider a token frame from node X'05' to node X'10'. The frame appears as:

Description	Value	CRC Register After Octet is Processed
preamble 1, not included in CRC	X'55'	
preamble 2, not included in CRC	X'FF'	
		X'FF' (initial value)
frame type = TOKEN	X'00'	X'55'
destination address	X'10'	X'C2'
source address	X'05'	X'BC'
data length MSB = 0	X'00'	X'95'
data length LSB = 0	X'00'	X'73' ones complement is X'8C'
Header CRC	X'8C'	X'55' final result at receiver

Thus, the transmitter would calculate the CRC on the five octets X'00', X'10', X'05', X'00', and X'00' to be X'73'. The ones complement of this is X'8C', which is appended to the frame.

The receiver would calculate the CRC on the six octets X'00', X'10', X'05', X'00', X'00', and X'8C' to be X'55', which is the expected result for a correctly received frame.

G.1.1 Sample Implementation of the Header CRC Algorithm in C

As an example, a C language implementation of the header CRC algorithm is presented. Inputs are the octet to be processed and the accumulating CRC value. The function return value is the updated CRC value.

In order to minimize shifting, we make use of the bits shifted left out of the least significant octet. In particular, bit 8 will be exclusive OR'ed with the least significant octet.

Desired result:

C7	C6	C5	C4	C3	C2	C1	C0
C6	C5	C4	C3	C2	C1	C0	C7
C5	C4	C3	C2	C1	C0		C6
C4	C3	C2	C1	C0			C5
C3	C2	C1	C0				C4
C2	C1	C0					C3
C1	C0						C2
C0							C1

Shifted 16 bit word:

								v
							C7	v<<1
						C7	C6	v<<2
					C7	C6	C5	v<<3
				C7	C6	C5	C4	v<<4
			C7	C6	C5	C4	C3	v<<5
		C7	C6	C5	C4	C3	C2	v<<6
C7	C6	C5	C4	C3	C2	C1		v<<7

C7	C6	C5	C4	C3	C2	C1		
C6	C5	C4	C3	C2	C1	C0		
C5	C4	C3	C2	C1	C0			
C4	C3	C2	C1	C0				
C3	C2	C1	C0					
C2	C1	C0						
C1	C0							
C0								

```
/*  Accumulate "dataValue" into the CRC in crcValue.
/   Return value is updated CRC
/
/   Assumes that "unsigned char" is equivalent to one octet.
/   Assumes that "unsigned int" is 16 bits.
/   The ^ operator means exclusive OR.
*/
unsigned char CalcHeaderCRC(unsigned char dataValue, unsigned char crcValue)
{
  unsigned int crc;

  crc = crcValue ^ dataValue; /* XOR C7..C0 with D7..D0 */
```

```
  /* Exclusive OR the terms in the table (top down) */
  crc = crc ^ (crc << 1) ^ (crc << 2) ^ (crc << 3)
            ^ (crc << 4) ^ (crc << 5) ^ (crc << 6) ^ (crc << 7);

  /* Combine bits shifted out left hand end */
  return (crc & 0xfe) ^ ((crc >> 8) & 1);
}
```

G.1.2 Sample Implementation of the Header CRC Algorithm in Assembly Language

As an example, an assembly language implementation of the Header CRC algorithm for the 68HC11 (an eight bit processor with a simple instruction set) is presented. The routine accumulates the CRC into the variable CRCLO. T1 is a temporary storage variable. Most instructions act on either accumulator A or B.

```
HEADERCRC:                ;ACCUMULATE THE OCTET (0,X) INTO THE DATA CRC
      LDAB  0,X           ;FETCH DATA OCTET (INDEXED OPERATION)
      EORB  CRCLO         ;D7-D0 EXCLUSIVE OR C7-C0
      STAB  CRCLO         ;SAVE RESULT
      CLRA                ;CLEAR REGISTER A
      ASLD                ;SHIFT (A,B) LEFT AS A 16 BIT VALUE
      ; A=(- - - - - - - 7) B=( 6 5 4 3 2 1 0 -)
      EORB  CRCLO
      ASLD
      ; A=(- - - - - - 7 6) B=( 5 4 3 2 1 0 - -)
      ;    (- - - - - - - 7) B=( 6 5 4 3 2 1 0 -)
      EORB  CRCLO
      ASLD
      ; A=(- - - - - 7 6 5) B=( 4 3 2 1 0 - - -)
      ;    (- - - - - - 7 6) B=( 5 4 3 2 1 0 - -)
      ;    (- - - - - - - 7) B=( 6 5 4 3 2 1 0 -)
      EORB  CRCLO
      ASLD
      ; A=(- - - - 7 6 5 4) B=( 3 2 1 0 - - - -)
      ;    (- - - - - 7 6 5) B=( 4 3 2 1 0 - - -)
      ;    (- - - - - - 7 6) B=( 5 4 3 2 1 0 - -)
      ;    (- - - - - - - 7) B=( 6 5 4 3 2 1 0 -)
      EORB  CRCLO
      ASLD
      ; A=(- - - 7 6 5 4 3) B=( 2 1 0 - - - - -)
      ;    (- - - - 7 6 5 4) B=( 3 2 1 0 - - - -)
      ;    (- - - - - 7 6 5) B=( 4 3 2 1 0 - - -)
      ;    (- - - - - - 7 6) B=( 5 4 3 2 1 0 - -)
      ;    (- - - - - - - 7) B=( 6 5 4 3 2 1 0 -)
      EORB  CRCLO
      ASLD
      ; A=(- - 7 6 5 4 3 2) B=( 1 0 - - - - - -)
      ;    (- - - 7 6 5 4 3) B=( 2 1 0 - - - - -)
      ;    (- - - - 7 6 5 4) B=( 3 2 1 0 - - - -)
      ;    (- - - - - 7 6 5) B=( 4 3 2 1 0 - - -)
      ;    (- - - - - - 7 6) B=( 5 4 3 2 1 0 - -)
      ;    (- - - - - - - 7) B=( 6 5 4 3 2 1 0 -)
      EORB  CRCLO
      ASLD
      ; A=(- 7 6 5 4 3 2 1) B=( 0 - - - - - - -)
      ;    (- - 7 6 5 4 3 2) B=( 1 0 - - - - - -)
      ;    (- - - 7 6 5 4 3) B=( 2 1 0 - - - - -)
      ;    (- - - - 7 6 5 4) B=( 3 2 1 0 - - - -)
      ;    (- - - - - 7 6 5) B=( 4 3 2 1 0 - - -)
      ;    (- - - - - - 7 6) B=( 5 4 3 2 1 0 - -)
      ;    (- - - - - - - 7) B=( 6 5 4 3 2 1 0 -)
      EORB  CRCLO
```

```
ANDB   #0FEH          ;CLEAR LSB OF BOTTOM HALF
ANDA   #01H           ;CLEAR ALL BUT LSB OF HIGH HALF
STAA   T1
; A=(- - - - - - - - 1) B=( 0 - - - - - - - -)
;    (- - - - - - - - 2) B=( 1 0 - - - - - - -)
;    (- - - - - - - - 3) B=( 2 1 0 - - - - - -)
;    (- - - - - - - - 4) B=( 3 2 1 0 - - - - -)
;    (- - - - - - - - 5) B=( 4 3 2 1 0 - - - -)
;    (- - - - - - - - 6) B=( 5 4 3 2 1 0 - - -)
;    (- - - - - - - - 7) B=( 6 5 4 3 2 1 0 - -)
;    (- - - - - - - - -) B=( 7 6 5 4 3 2 1 - -)
EORB   T1             ;COMBINE LSB OF HIGH HALF
STAB   CRCLO
RTS
```

G.1.3 Other Implementations of the Header CRC Algorithm

Other implementations of the Header CRC algorithm are possible. It may be seen that the new CRC value is a function of the prior CRC value exclusive OR'ed with the data value. Thus, a lookup table with 256 elements may be used to quickly determine the new CRC value, using the prior CRC exclusive OR'ed with the data value as an index. The contents of the table may be computed using the implementations shown in G.1.1 or G.1.2.

G.2 Calculation of the Data CRC

We begin with the diagram of a hardware CRC generator as shown in Figure G-2. The polynomial used is the CRC-CCITT:

$$X^{16} + X^{12} + X^5 + 1$$

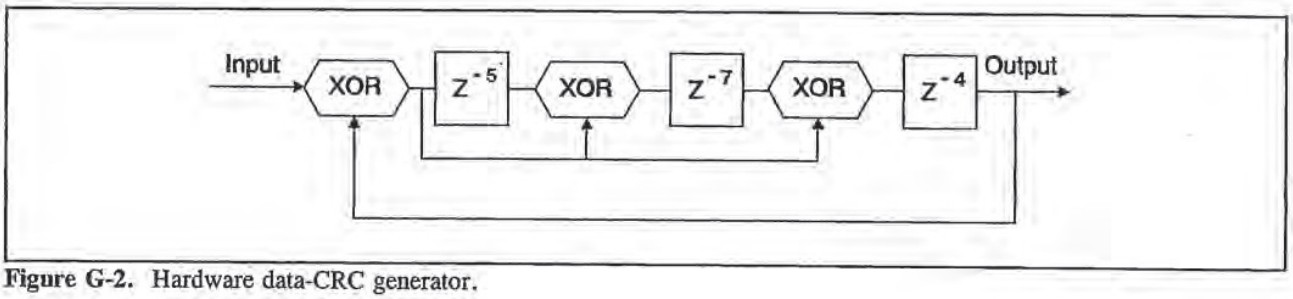

Figure G-2. Hardware data-CRC generator.

The hardware implementation operates on a serial bit stream, whereas our calculation must operate on entire octets. To this end, we follow the operation of the circuit through eight bits of data. The CRC shift register is initialized to X0 (input end) to X15 (output end), the input data is D0 to D7 (least significant bit first, as transmitted and received by a UART). Within each block below, the terms are exclusive OR'ed vertically.

input	register contents															
D0	X0	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15
D1		X0	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14
	D0					D0							D0			
	X15					X15							X15			
D2			X0	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13
	D1	D0				D1	D0						D1	D0		
	X14	X15				X14	X15						X14	X15		
D3				X0	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12
	D2	D1	D0			D2	D1	D0					D2	D1	D0	
	X13	X14	X15			X13	X14	X15					X13	X14	X15	
D4				X0	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	
	D3	D2	D1	D0		D3	D2	D1	D0				D3	D2	D1	D0
	X12	X13	X14	X15		X12	X13	X14	X15				X12	X13	X14	X15
D5					X0	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	
		D3	D2	D1	D0	D3	D2	D1	D0					D3	D2	D1
		X12	X13	X14	X15		X12	X13	X14	X15				X12	X13	X14
	D4					D4				D4						
	X11					X11				X11						
	D0					D0				D0						
	X15					X15				X15						
D6							X0	X1	X2	X3	X4	X5	X6	X7	X8	X9
			D3	D2	D1	D0		D3	D2	D1	D0				D3	D2
			X12	X13	X14	X15		X12	X13	X14	X15				X12	X13
	D5	D4				D5	D4						D5	D4		
	X10	X11				X10	X11						X10	X11		
	D1	D0				D1	D0						D1	D0		
	X14	X15				X14	X15						X14	X15		
D7								X0	X1	X2	X3	X4	X5	X6	X7	X8
				D3	D2	D1	D0		D3	D2	D1	D0				D3
				X12	X13	X14	X15		X12	X13	X14	X15				X12
	D6	D5	D4			D6	D5	D4					D6	D5	D4	
	X9	X10	X11			X9	X10	X11					X9	X10	X11	
	D2	D1	D0			D2	D1	D0					D2	D1	D0	
	X13	X14	X15			X13	X14	X15					X13	X14	X15	
									X0	X1	X2	X3	X4	X5	X6	X7
				D3	D2	D1	D0			D3	D2	D1	D0			
				X12	X13	X14	X15			X12	X13	X14	X15			
	D7	D6	D5	D4		D7	D6	D5	D4				D7	D6	D5	D4
	X8	X9	X10	X11		X8	X9	X10	X11				X8	X9	X10	X11
	D3	D2	D1	D0		D3	D2	D1	D0				D3	D2	D1	D0
	X12	X13	X14	X15		X12	X13	X14	X15				X12	X13	X14	X15

The final block above is the net result of shifting an octet of data through the CRC generator. By performing the indicated exclusive OR operations, an octet can be processed into the CRC. In order to simplify the required shifting operations, we define X0 to be the most significant bit of the parallel representation and X15 to be the least significant. Since most microprocessors number bits in the opposite order, we rename the bits to correspond to standard microprocessor bit nomenclature. Let C15 = X0, C14 = X1, etc. The final block in the previous table thus becomes:

```
|-------------------------------------------------------------------------------|
| X0  X1  X2  X3  X4  X5  X6  X7  X8  X9  X10 X11 X12 X13 X14 X15               |
|-------------------------------+-----------------------------------------------|
| C15 C14 C13 C12 C11 C10 C9  C8  C7  C6  C5  C4  C3  C2  C1  C0                |
|-------------------------------+-----------------------------------------------|
|                               |     C15 C14 C13 C12 C11 C10 C9  C8            |
|                 D3  D2  D1  D0|         D3  D2  D1  D0                        |
|                 C3  C2  C1  C0|         C3  C2  C1  C0                        |
| D7  D6  D5  D4      D7  D6  D5  D4                  D7  D6  D5  D4            |
| C7  C6  C5  C4      C7  C6  C5  C4                  C7  C6  C5  C4            |
| D3  D2  D1  D0      D3  D2  D1  D0                  D3  D2  D1  D0            |
| C3  C2  C1  C0      C3  C2  C1  C0                  C3  C2  C1  C0            |
|-------------------------------+-----------------------------------------------|
```

or, rearranging vertically (since exclusive OR is commutative) to minimize computation:

```
|-------------------------------+-----------------------------------------------|
|                               |     C15 C14 C13 C12 C11 C10 C9  C8            |
| D7  D6  D5  D4  D3  D2  D1  D0|                                               |
| C7  C6  C5  C4  C3  C2  C1  C0|                                               |
|                 D7  D6  D5  D4|  D3  D2  D1  D0                               |
|                 C7  C6  C5  C4|  C3  C2  C1  C0                               |
| D3  D2  D1  D0                |                 D7  D6  D5  D4                |
| C3  C2  C1  C0                |                 C7  C6  C5  C4                |
|                               |                 D3  D2  D1  D0                |
|                               |                 C3  C2  C1  C0                |
|                 D3  D2  D1  D0|                                               |
|                 C3  C2  C1  C0|                                               |
|-------------------------------+-----------------------------------------------|
```

In operation, the CRC register C15-C0 is initialized to all ones. During transmission, the Data are passed though the calculation before transmission. The ones complement of the final CRC register value is then transmitted with the least significant octet (C7-C0) first. At the receiving end, all data octets, including the received CRC octets, are passed though the calculation. If all octets are received correctly, then the final value of the receiver's CRC register will be the constant X'F0B8':

```
|  1   1   1   1   0   0   0   0   1   0   1   1   1   0   0   0  |
|<------ F -----><------ 0 -----><------ B -----><------ 8 ------>|
```

As an example of usage, consider a data sequence X'01', X'22', X'30':

Description	Value	CRC Accumulator After Octet is Processed
		X'FFFF' (initial value)
first data octet	X'01'	X'1E0E'
second data octet	X'22'	X'EB70'
third data octet	X'30'	X'42EF' ones complement is X'BD10'
CRC1 (least significant octet)	X'10'	X'0F3A'
CRC2 (most significant octet)	X'BD'	X'F0B8' final result at receiver

Thus, the transmitter would calculate the CRC on the three octets X'01', X'22', and X'30' to be X'42EF'. The ones complement of this is X'BD10', which is appended to the frame least significant octet first as X'10', X'BD'.

The receiver would calculate the CRC on the five octets X'01', X'22', X'30', X'10', and X'BD' to be X'F0B8', which is the expected result for a correctly received frame.

G.2.1 Sample Implementation of the Data CRC Algorithm in C

As an example, a C language implementation of the Data CRC algorithm is presented. Inputs are the octet to be processed and the accumulating CRC value. The function return value is the updated CRC value.

```
/*  Accumulate "dataValue" into the CRC in crcValue.
/   Return value is updated CRC
/
/   Assumes that "unsigned char" is equivalent to one octet.
/   Assumes that "unsigned int" is 16 bits.
/   The ^ operator means exclusive OR.
*/

unsigned int CalcDataCRC(unsigned char dataValue, unsigned int crcValue)
{
  unsigned int crcLow;

  crcLow = (crcValue & 0xff) ^ dataValue; /* XOR C7..C0 with D7..D0 */

  /* Exclusive OR the terms in the table (top down) */
  return (crcValue >>8) ^ (crcLow << 8)    ^ (crcLow <<3)
                        ^ (crcLow <<12)    ^ (crcLow >> 4)
                        ^ (crcLow & 0x0f) ^ ((crcLow & 0x0f) << 7);
}
```

G.2.2 Sample Implementation of the Data CRC Algorithm in Assembly Language

As an example, an assembly language implementation of the Data CRC calculation for the 68HC11 (an eight-bit processor with a simple instruction set) is presented. The routine accumulates the CRC into the variables CRCLO and CRCHI. T1 and T2 are temporary storage variables. Most instructions act on either accumulator A or B.

```
DATACRC:                ;ACCUMULATE THE OCTET (0,X) INTO THE DATA CRC
      LDAA  0,X         ;FETCH DATA OCTET (INDEXED OPERATION)
      EORA  CRCLO       ;D7-D0 EXCLUSIVE OR C7-C0
      STAA  CRCLO       ;SAVE RESULT
      CLRB              ;CLEAR REGISTER B
      LSRA              ;SHIFT A RIGHT, 0 INTO MSB, LSB INTO CARRY
      RORB              ;ROTATE B RIGHT, CARRY INTO MSB
      LSRA
      RORB
      LSRA
      RORB
      LSRA
      RORB
      ; A=(- - - - 7 6 5 4) B=( 3  2  1  0  -  -  -  -)
      ;
      EORA  CRCLO
      ANDA  #0FH        ;MASK OFF ALL BUT LS 4 BITS
      ; A=(- - - - 7 6 5 4) B=( 3  2  1  0  -  -  -  -)
      ;    (- - - - 3 2 1 0)
      ;
      STAA  T1          ;SAVE TEMP RESULT (NOT USED EXCEPT AS TEMP)
      STAB  T2
      LSRA              ;SHIFT RIGHT 1 BIT
      RORB
      ; A=(- - - - - 7 6 5) B=( 4  3  2  1  0  -  -  -)
      ;    (- - - - - 3 2 1)    ( 0  -  -  -  -  -  -  -)
      ;
      EORA  CRCLO       ;COMBINE PARTIAL TERMS
```

```
EORA  T2
EORB  CRCHI         ;C8-C15
EORB  T1
; A=(- - - - - 7 6 5) B=( 4  3  2  1  0  -  -  -)
;    (- - - - - 3 2 1)    ( 0  -  -  -  -  -  -  -)
;    (7 6 5 4 3 2 1 0)    (15 14 13 12 11 10  9  8)
;    (3 2 1 0 - - - -)    ( -  -  -  -  7  6  5  4)
;                         ( -  -  -  -  3  2  1  0)
;
STAA  CRCHI         ;SAVE RESULT
STAB  CRCLO
RTS
```

G.2.3 Other Implementations of the Data CRC Algorithm

Other implementations of the Data CRC algorithm are possible. It can be seen that the new CRC value may be factored into the exclusive OR of two terms. One term is the most significant octet of the prior CRC value. The other term is a function of the least significant octet of the prior CRC value exclusive OR'ed with the data value. A lookup table with 256 elements may be used to quickly determine the value of the second term, using the least significant octet of the prior CRC exclusive OR'ed with the data value as an index. This term may then be exclusive OR'ed with the most significant octet of the prior CRC value to form the new CRC value. The contents of the table may be computed using the implementation shown in G.2.1 or G.2.2.

ANNEX H - COMBINING BACnet NETWORKS WITH NON-BACnet NETWORKS (NORMATIVE)

(This annex is part of this standard and is required for its use.)

H.1 Mapping Non-BACnet Networks onto BACnet Routers

In addition to providing the means to interconnect multiple BACnet networks, BACnet routers may also be used to provide a gateway function to non-BACnet networks. Non-BACnet networks are characterized by the use of message structures, procedures, and medium access control techniques other than those contained in this standard. The mapping from BACnet to non-BACnet networks is performed by extending the routing table concept to allow non-BACnet devices to be addressable using BACnet NPCI. Thus, each non-BACnet network is assigned a unique two-octet network number, and each device on the non-BACnet network is represented by a "MAC address" that may, or may not, correspond to the actual octets used to address the device using the medium access control in use on the foreign network. Since communication with devices on non-BACnet networks is, by definition, not standardized here, the specific procedures for interpreting, translating, or relaying messages received by such a router-gateway from either the BACnet or non-BACnet ports are outside the scope of this standard.

H.2 Multiple "Virtual" BACnet Devices in a Single Physical Device

A BACnet device is one that possesses a Device object and communicates using the procedures specified in this standard. In somes instances, however, it may be desirable to model the activities of a physical building automation and control device through the use of more than one BACnet device. Each such device will be referred to as a "virtual BACnet device." This can be accomplished by configuring the physical device to act as a router to one or more "virtual networks." The idea is that each virtual BACnet device would be associated with a unique DNET and DADR pair, i.e., a unique BACnet address. The physical device would be expected to perform exactly as if it were a router between the network on which messages for the virtual BACnet devices are received and a real BACnet network with the same network number as assigned to the virtual BACnet devices.

H.3 Using BACnet with the DARPA Internet Protocols

This subclause describes procedures whereby BACnet messages may be conveyed using the protocols developed by the Defense Advanced Research Projects Agency (DARPA) of the Department of Defense (DoD). Collectively, these protocols are known as the Internet Protocol suite. The methodology described in this appendix involves encapsulating/decapsulating BACnet LSDUs and conveying them across an internet using the capabilities of IP routers, a technique often referred to as "tunneling." Although this appendix will describe the procedures in terms of a BACnet/Internet Protocol Packet-Assembler-Disassembler, the necessary functionality could be built into BACnet nodes directly.

H.3.1 BACnet/Internet Protocol Packet-Assembler-Disassemblers (B/IP PAD)

A B/IP PAD is a device that implements the BACnet network layer as described in Clause 6 of this standard as well as the DoD User Datagram Protocol (UDP) and the Internet Protocol (IP).

Upon receipt of a BACnet packet from the local network, the B/IP PAD inspects the NPCI to see if a DNET network number has been supplied. If so, the B/IP PAD then consults an internal table consisting of entries mapping network numbers, IP addresses of all B/IP PADs within the BACnet internetwork, and the IP address of an IP router on the local network that represents the next hop for the IP datagram. If an appropriate entry is found, the B/IP PAD encapsulates the LSDU portion of the BACnet message (see Figure H-1) in a UDP packet where the LSDU represents the data portion of the packet. The UDP source and destinations ports shall be set to X'BAC0'. The B/IP PAD shall then send an IP datagram containing the UDP packet to the local IP router indicating its own IP address as the source address and the IP address of the B/IP PAD corresponding to the DNET contained in the BACnet message as the destination IP address. If the DNET in the BACnet message is the global broadcast address, this procedure shall result in the transmission of the BACnet LSDU to all B/IP PADs contained in the table. Conveyance of packets between B/IP PADs follows standard IP procedures.

Upon receipt of an IP datagram from an IP router, a B/IP PAD shall locate the BACnet process at UDP port X'BAC0', which shall prepare the data portion of the UDP datagram for transmission as a BACnet message on the local network using the procedures defined in 6.3.

H.3.2 Implementation Notes

An implementation on an 8802-3 LAN might be configured as shown in Figure H-1. BACnet packets are differentiated from IP packets and those of other protocols by the LSAP contained in the LLC header. IP uses an LSAP of X'06', whereas the BACnet network layer is identified by an LSAP of X'82'. In the case shown in the Figure, each packet actually appears twice on the network, once as a BACnet message and once as an IP message. B/IP PADs could also be constructed to implement the IP routing procedure for any IP packet in addition to performing the BACnet encapsulation/decapsulation. Such a device would be a B/IP PAD/Router. This is illustrated in Figure H-2.

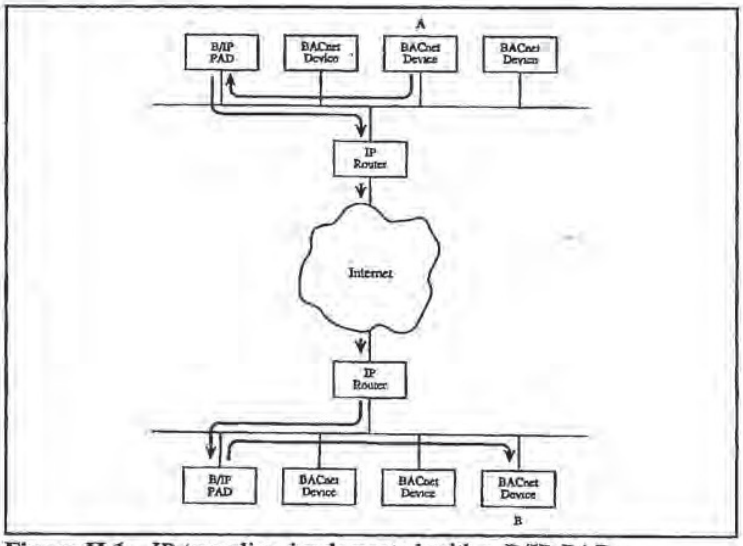

Figure H-1. IP tunneling implemented with a B/IP PAD.

For further information about the DoD protocols, consult the *DDN Protocol Handbook* referenced in Clause 25.

H.4 Using BACnet with the IPX Protocol

This subclause describes procedures whereby BACnet messages may be conveyed using the protocols developed by Novell Corporation, which are known as Internetwork Datagram Protocol or, more popularly, IPX. This protocol layer is based on a subset of the Xerox Network Services (XNS) protocols, in particular using the Packet Exchange Protocol (PEP). The methodology described in this subclause involves encapsulating/decapsulating BACnet LSDUs and conveying them across an IPX internetwork using the capabilities of IPX routers, a technique that here will be called *IPX Tunneling*. Although this appendix will describe the procedures in terms of a BACnet/IPX packet-assembler-disassembler, the necessary functionality could be built into BACnet nodes directly.

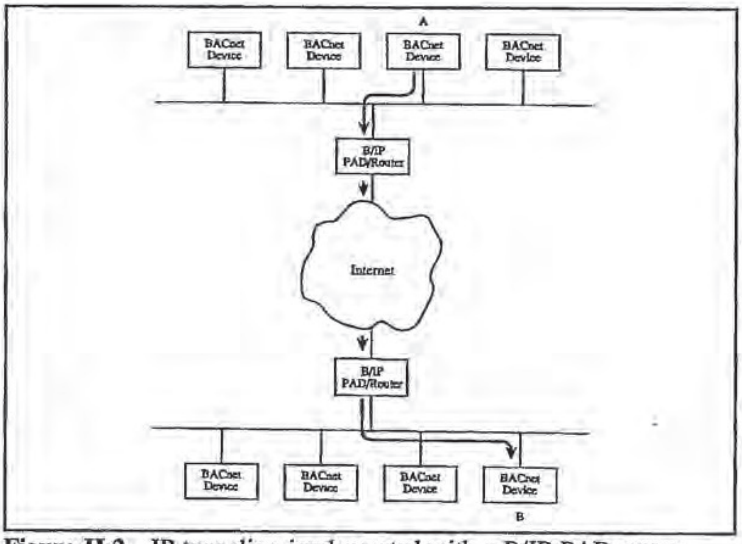

Figure H-2. IP tunneling implemented with a B/IP PAD router.

H.4.1 BACnet/IPX Packet-Assembler-Disassemblers (B/IPX PAD)

A B/IPX PAD is a device that implements both the BACnet network layer as described in Clause 6 of this standard, as well as the IPX network layer.

Upon receipt of a BACnet packet from the local network, the B/IPX PAD inspects the NPCI to see if a DNET network number has been supplied. If so, the B/IPX PAD then consults an internal table consisting of entries mapping BACnet network numbers to (IPX network number, MAC layer address) pairs that represent all remote B/IPX PADs within the BACnet internetwork. If an appropriate entry is found, the B/IPX PAD encapsulates the LSDU portion of the BACnet message (see Figure 6-1) in an IPX Tunnel packet (see Figure H-3) where

the destination network and MAC-layer addresses correspond to those of the remote B/IPX PAD and the destination socket number shall be set to the BACnet IPX socket number X'87C1'. The B/IPX PAD shall then send an IPX datagram containing this tunnel packet to the local IPX router, indicating its own IPX network number, MAC-layer address, and the BACnet IPX socket number as the source. If the DNET in the BACnet message is the global broadcast address, this procedure shall result in the transmission of the BACnet LSDU tunnel packet to all B/IPX PADs contained in the table. Conveyance of packets between B/IPX PADs follows standard IPX routing procedures.

Upon receipt of an IPX datagram using the BACnet socket, a B/IPX PAD shall prepare the data portion of the BACnet IPX Tunnel packet for transmission as a BACnet message on the local network using the procedures defined in 6.3.

```
;IPX Header (30 octets)
        dw      X'FFFF'         ;XNS checksum (not used by IPX)
        dw      ?               ;packet length (set by IPX)
        db      ?               ;routers crossed (set by IPX)
        db      4               ;type 4=Packet Exchange Packet
        dd      ?               ;32 bit dest network number
        db      ?,?,?,?,?,?     ;dest MAC layer address
        dw      X'87C1'         ;dest BACnet socket
        dd      ?               ;32 bit source network number
        db      ?,?,?,?,?,?     ;source MAC layer address
        dw      X'87C1'         ;source BACnet socket
        db 546 dup ?            ;BACnet LSDU
```

Figure H-3. Structure of a BACnet IPX tunnel packet.

H.4.2 Implementation Notes

An implementation on an 8802-3 LAN might be configured as shown in Figure H-1. BACnet packets are differentiated from IPX packets and those of other protocols by the LSAP contained in the LLC header. IPX uses an LSAP of X'E0', whereas the BACnet network layer is identified by an LSAP of X'82'. In the case shown in the Figure, each packet actually appears twice on the network: once as a BACnet message and once as an IPX message. B/IPX PADs could also be constructed to implement the IPX routing procedure for any IPX packet in addition to perfoming the BACnet encapsulation/decapsulation. Such a device would be a B/IPX PAD/Router. This is illustrated in Figure H-2.

References

NetWare® System Interface Technical Overview
Addison-Wesley Publishing Company, Reading Massachusetts
ISBN 0-201-57027-0

ANNEX I - COMMANDABLE PROPERTIES WITH MINIMUM ON AND OFF TIMES (INFORMATIVE)

(This annex is not part of this standard but is included for information purposes only)

This annex is an example of the use of a commandable property with minimum on and off times.

Suppose that we have a binary output object with a 600-second (10-minute) Minimum_On_Time, a 1200-second (20-minute) Minimum_Off_Time, and NORMAL polarity. The Priority_Array is all NULL, and the Present_Value is INACTIVE (off) due to the Relinquish_Default and has been INACTIVE for several hours. See Figure I-1(a).

In Figure I-1(b) a write is made to the Present_Value at priority 9 with a value of ACTIVE (on). Logic internal to the request server writes the value to entry 9 in the Priority_Array. Since the Present_Value has been off for more than 20 minutes, the change of state can occur immediately. Thus, the Present_Value will take the value ACTIVE and the Change_Of_State_Time will be set to the time of control.

Because the state of the Present_Value has changed, the minimum on and off time maintenance entity writes the new state of ACTIVE to entry 6 in the Priority_Array in order to enforce the minimum time. Any further writes to the Present_Value at priorities less than 6 will be entered into the Priority_Array but will not be acted upon due to the presence of the ACTIVE at priority 6. For example, in Figure I-1(c), a write to Present Value at priority 7 with a value of INACTIVE will be entered into the Priority_Array but will not be acted upon.

Writes to priorities higher than 6 will be entered into the Priority_Array and may cause changes of state due to their higher priority. This is desired for emergency and fire control.

In Figure I-1(e) the minimum on and off time maintenance entity in the device issues a relinquish (NULL) at priority 6 when the Minimum_On_Time expires, 10 minutes after the initial write. The remaining entries in the Priority_Array are then examined to determine the new Present_Value. If no change of state results, then no further action occurs. This would be the case in our example if no INACTIVE requests above priority 9 had been made.

In our example we had an INACTIVE at priority 7. Thus, in Figure I-1(f), when the Minimum_On_Time expires and the ACTIVE at priority 6 is relinquished, the value at priority 7 takes control. The Present_Value changes state to INACTIVE. As before, because the state of the Present_Value has changed, the minimum on and off time maintenance entity writes the new state of INACTIVE to entry 6 in the Priority_Array in order to enforce the minimum time.

We note that while write to priorities above 6 are not subject to minimum on and off times, if such writes cause changes of states, the server should still write the new value to priority 6. Thus, if the higher priority request is relinquished within the minimum time, the minimum will be enforced before any lower priority requests can cause changes of state.

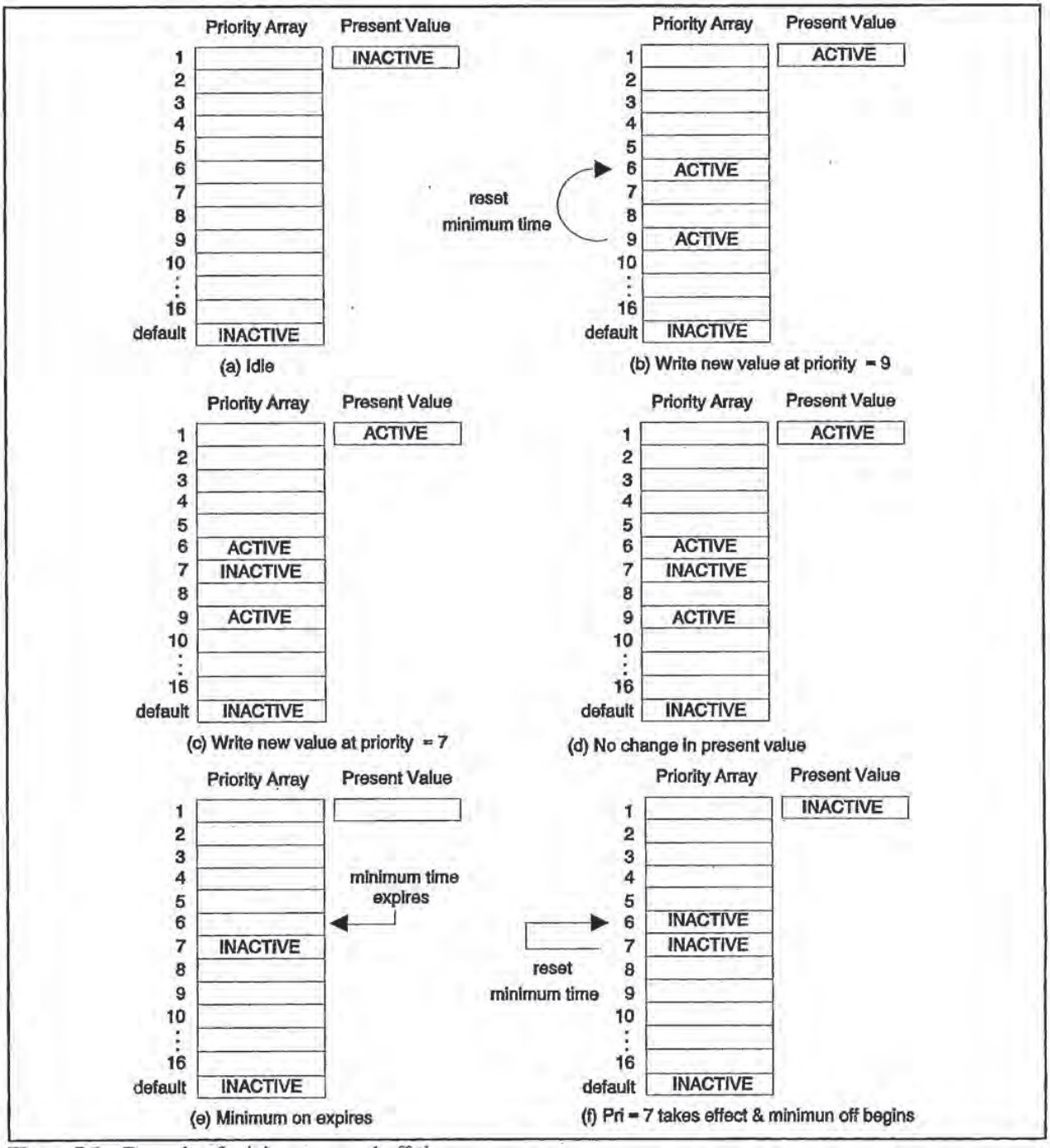

Figure I-1. Example of minimum on and off times.

Addendum *a* to ANSI/ASHRAE 135-1995

ASHRAE® STANDARD

BACnet® - A Data Communication Protocol for Building Automation and Control Networks

Approved by the ASHRAE Standards Committee January 23, 1999; by the ASHRAE Board of Directors January 27, 1999; and by the American National Standards Institute October 1, 1999.

This standard is under continuous maintenance by a Standing Standard Project Committee (SSPC) for which the Standards Committee has established a documented program for regular publication of addenda or revisions, including procedures for timely, documented, consensus action on requests for change to any part of the standard. The change submittal form, instructions, and deadlines are given at the back of this standard and may be obtained in electronic form from ASHRAE's Internet Home Page, *http://www/ashrae.org*, or in paper form from the Manager of Standards. The latest edition of an ASHRAE Standard and printed copies of a public review draft may be purchased from ASHRAE Customer Service, 1791 Tullie Circle, NE, Atlanta, GA 30329-2305. E-mail: *orders@ashrae.org*. Fax: 404-321-5478. Telephone: 404-636-8400 (worldwide), or toll free 1-800-527-4723 (for orders in U.S. and Canada).

ISSN 1041-2336

AMERICAN SOCIETY OF HEATING, REFRIGERATING AND AIR-CONDITIONING ENGINEERS, INC.
1791 Tullie Circle, NE • Atlanta, GA 30329

ASHRAE STANDING STANDARD PROJECT COMMITTEE 135
Cognizant TC: TC 1.4, Control Theory and Application
SPLS Liaison: Bruce A. Wilcox

*H. Michael Newman, *Chairman*
*Steven T. Bushby, *Vice-Chairman*
*Ron E Anderson
*Beauford W. Atwater
*Keith A. Corbett
*Jeffrey Cosiol
*Harsha M. Dabholkar
*David M. Fisher
*Ira G Goldschmidt
*Anthony J. Icenhour
*Jerald P. Martocci
*Cherisse M. Nicastro
*Robert L. Old, Jr.
*David Robin
*Patrick Sheridan
*William O. Swan, III, *Secretary*
*Grant N. Wichenko
*Robert J. Zamojcin
James F. Butler
Dana R. Epperson
Jerald Griliches
John L. Hartman
Winston I Hetherington
Richard Holtz
Carl Neilson
Kevin G. Sweeney

**Denotes members of voting status when this standard was approved for publication*

ASHRAE STANDARDS COMMITTEE JANUARY 1999

Michael R. Bilderbeck, *Chairman*
Arthur E McIvor, *Vice-Chairman*
George F. Carscallen
Waller S. Clements
Piotr A. Domanski
Richard A. Evans
Mark C. Hegberg
Martha J. Hewett
Douglas C. Hittle
Frederick H. Kohloss
William J. Landman
Rodney H. Lewis
Nance C. Lovvorn
Amanda Meitz
Davor Novosel
Joseph A. Pietsch
James A. Ranfone
Gaylon Richardson
Ganesan Sundaresan
Thomas E Watson
Bruce A. Wilcox
J Richard Wright
James E Woods, BOD ExO
Ronald P. Vallort, CO

Claire Ramspeck, Manager of Standards

SPECIAL NOTE

This American National Standard (ANS) is a national voluntary consensus standard developed under the auspices of the American Society of Heating, Refrigerating, and Air-Conditioning Engineers (ASHRAE) Consensus is defined by the American National Standards Institute (ANSI), of which ASHRAE is a member and which has approved this standard as an ANS, as "substantial agreement reached by directly and materially affected interest categories This signifies the concurrence of more than a simple majority, but not necessarily unanimity Consensus requires that all views and objections be considered, and that an effort be made toward their resolution " Compliance with this standard is voluntary until and unless a legal jurisdiction makes compliance mandatory through legislation

ASHRAE obtains consensus through participation of its national and international members, associated societies, and public review.

ASHRAE Standards are prepared by a Project Committee appointed specifically for the purpose of writing the Standard The Project Committee Chair and Vice-Chair must be members of ASHRAE, while other committee members may or may not be ASHRAE members, all must be technically qualified in the subject area of the Standard Every effort is made to balance the concerned interests on all Project Committees.

The Manager of Standards of ASHRAE should be contacted for:

a interpretation of the contents of this Standard,
b participation in the next review of the Standard,
c. offering constructive criticism for improving the Standard,
d permission to reprint portions of the Standard

ASHRAE INDUSTRIAL ADVERTISING POLICY

ASHRAE Standards and Guidelines are established to assist industry and the public by offering a uniform method of testing for rating purposes, by suggesting safe practices in designing and installing equipment, by providing proper definitions of this equipment, and by providing other information that may serve to guide the industry. The creation of ASHRAE Standards and Guidelines is determined by the need for them, and conformance to them is completely voluntary.

In referring to this Standard or Guideline and in marketing of equipment and in advertising, no claim shall be made, either stated or implied, that the product has been approved by ASHRAE

DISCLAIMER

ASHRAE uses its best efforts to promulgate Standards and Guidelines for the benefit of the public in light of available information and accepted industry practices However, ASHRAE does not guarantee, certify, or assure the safety or performance of any products, components, or systems tested, installed, or operated in accordance with ASHRAE's Standards or Guidelines or that any tests conducted under its Standards or Guidelines will be nonhazardous or free from risk.

[Add these abbreviations and acronyms to 3.3]

BBMD	BACnet Broadcast Management Device
BVLL	BACnet Virtual Link Layer
BVLC	BACnet Virtual Link Control
BVLCI	BACnet Virtual Link Control Information
BDT	Broadcast Distribution Table
B/IP	BACnet/IP
B/IP-M	BACnet/IP Multicast
FDT	Foreign Device Table
IP	Internet Protocol - RFC 791
PPP	Point-to-Point Protocol - RFC 1661
UDP	User Datagram Protocol - RFC 768
RFC	Request for Comment
SLIP	Serial Line Internet Protocol - RFC 1055

[Add the following entry to Table 6-1; p. 50]

Data Link Technology	Maximum NPDU Length
BACnet/IP as defined in Annex J	1497 octets
...	...

[Add the following Annex]

Annex J - BACnet/IP

J.1 General

This normative annex specifies the use of BACnet messaging with the networking protocols originally defined as the result of reseach sponsored by the U. S. government's Defense Advanced Research Projects Agency and now maintained by the Internet Engineering Task Force. This suite of protocols is generally known as the "Internet Protocols."

J.1.1 BACnet/IP (B/IP) Network Definition

A BACnet/IP network is a collection of one or more IP subnetworks (IP domains) that are assigned a single BACnet network number. A BACnet internetwork (3.2.26) consists of two or more BACnet networks. These networks may be BACnet/IP networks or use the technologies specified in Clauses 7, 8, 9, and 11. This standard also supports the inclusion of IP multicast groups in a fashion analogous to IP subnets, as described below in J.8.

J.1.2 Addressing within B/IP Networks

In the case of B/IP networks, six octets consisting of the four-octet IP address followed by a two-octet UDP port number (both of which shall be transmitted most significant octet first) shall function analogously to the MAC address of the technologies of Clauses 7, 8, 9, and 11 with respect to communication between individual devices and inclusion in the Clause 6 NPCI, where a DADR or SADR is required. This address shall be referred to as a B/IP address. The default UDP port for both directed messages and broadcasts shall be X'BAC0' and all B/IP devices shall support it. In some cases, e.g., a situation where it is desirable for two groups of BACnet devices to coexist independently on the same IP subnet, the UDP port may be configured locally to a different value without it being considered a violation of this protocol. Where the "B/IP broadcast address" is referred to in this Annex, it means an IP address with the subnet of the broadcasting device in the network portion and all 1's in the host portion of the address and the UDP port of the devices on the B/IP network in question. An IP multicast address in conjunction with an appropriate UDP port may be used in lieu of the B/IP broadcast address under the circumstances defined in J.8.

J.1.3 B/IP Concept

A BACnet/IP network shall function in concept identically to the other non-IP network types with respect to directed messages and broadcast messages, including local, remote, and global broadcasts, as defined in 6.3.2: a directed message shall be sent directly to the destination node; a "local broadcast" shall reach all nodes on a single B/IP network; a "remote broadcast" shall reach all nodes on a single BACnet network with network number different from that of the originator's network; a "global broadcast" shall reach all nodes on all networks. The management of broadcasts within a single B/IP network, or between multiple B/IP networks, or between B/IP and non-B/IP networks, is described in J.4.

J.2 BACnet Virtual Link Layer

The BACnet Virtual Link Layer (BVLL) provides the interface between the BACnet Network Layer (Clause 6) and the underlying capabilities of a particular communication subsystem. This Annex specifies the BACnet Virtual Link Control (BVLC) functions required to support BACnet/IP directed and broadcast messages. The purpose and format of each message is described in the following subclauses.

Note that each BVLL message has at least three fields. The 1-octet BVLC Type field indicates which underlying communication subsystem or microprotocol is in use. In this case, a BVLC Type of X'81' indicates the use of BACnet/IP as defined in this Annex. The 1-octet BVLC Function field identifies the specific function to be carried out in support of the indicated communication subsystem or microprotocol type. The 2-octet BVLC Length field is the length, in octets, of the entire BVLL message, including the two octets of the length field itself, most significant octet first.

J.2.1 BVLC-Result: Purpose

This message provides a mechanism to acknowledge the result of those BVLL service requests that require an acknowledgment, whether successful (ACK) or unsuccessful (NAK). These are: Write-Broadcast-Distribution-Table (ACK, NAK); Read-Broadcast-Distribution-Table (NAK only); Register-Foreign-Device (ACK, NAK); Read-Foreign-Device-Table (NAK only); Delete-Foreign-Device-Table-Entry (ACK, NAK); and Distribute-Broadcast-To-Network (NAK only).

J.2.1.1 BVLC-Result: Format

The BVLC-Result message consists of four fields:

BVLC Type:	1-octet	X'81'	BVLL for BACnet/IP
BVLC Function:	1-octet	X'00'	BVLC-Result
BVLC Length:	2-octets	L	Length L, in octets, of the BVLL message
Result Code:	2-octets	X'0000'	Successful completion
		X'0010	Write-Broadcast-Distribution-Table NAK
		X'0020'	Read-Broadcast-Distribution-Table NAK
		X'0030'	Register-Foreign-Device NAK
		X'0040'	Read-Foreign-Device-Table NAK
		X'0050'	Delete-Foreign-Device-Table-Entry NAK
		X'0060'	Distribute-Broadcast-To-Network NAK

J.2.2 Write-Broadcast-Distribution-Table: Purpose

This message provides a mechanism for initializing or updating a Broadcast Distribution Table (BDT) in a BACnet Broadcast Management Device (BBMD).

J.2.2.1 Write-Broadcast-Distribution-Table: Format

The Write-Broadcast-Distribution-Table message consists of four fields:

BVLC Type:	1-octet	X'81'	BVLL for BACnet/IP
BVLC Function:	1-octet	X'01'	Write-Broadcast-Distribution-Table
BVLC Length:	2-octets	L	Length L, in octets, of the BVLL message

List of BDT Entries:	N*10-octets	

N indicates the number of entries in the BDT. Each BDT entry consists of the 6-octet B/IP address of a BBMD followed by a 4-octet field called the broadcast distribution mask that indicates how broadcast messages are to be distributed on the IP subnet served by the BBMD. See J.4.3.2.

J.2.3 Read-Broadcast-Distribution-Table: Purpose

The message provides a mechanism for retrieving the contents of a BBMD's BDT.

J.2.3.1 Read-Broadcast-Distribution-Table: Format

The Read-Broadcast-Distribution-Table message consists of three fields:

BVLC Type:	1-octet	X'81'	BVLL for BACnet/IP
BVLC Function:	1-octet	X'02'	Read-Broadcast-Distribution-Table
BVLC Length:	2-octets	X'0004'	Length, in octets, of the BVLL message

J.2.4 Read-Broadcast-Distribution-Table-Ack: Purpose

This message returns the current contents of a BBMD's BDT to the requester. An empty BDT shall be signified by a list of length zero.

J.2.4.1 Read-Broadcast-Distribution-Table-Ack: Format

The Read-Broadcast-Distribution-Table-Ack message consists of four fields:

BVLC Type:	1-octet	X'81'	BVLL for BACnet/IP
BVLC Function:	1-octet	X'03'	Read-Broadcast-Distribution-Table-Ack
BVLC Length:	2-octets	L	Length L, in octets, of the BVLL message
List of BDT Entries:	N*10-octets		

N indicates the number of entries in the BDT whose contents are being returned.

J.2.5 Forwarded-NPDU: Purpose

This BVLL message is used in broadcast messages from a BBMD as well as in messages forwarded to registered foreign devices. It contains the source address of the original node as well as the original BACnet NPDU.

J.2.5.1 Forwarded-NPDU: Format

The Forwarded-NPDU message consists of five fields:

BVLC Type:	1-octet	X'81'	BVLL for BACnet/IP
BVLC Function:	1-octet	X'04'	Forwarded-NPDU
BVLC Length:	2-octets	L	Length L, in octets, of the BVLL message
B/IP Address of Originating Device:			6-octets
BACnet NPDU from Originating Device:			Variable length

J.2.6 Register-Foreign-Device: Purpose

This message allows a foreign device, as defined in J.5.1, to register with a BBMD for the purpose of receiving broadcast messages.

J.2.6.1 Register-Foreign-Device: Format

The Register-Foreign-Device message consists of four fields:

BVLC Type:	1-octet	X'81'	BVLL for BACnet/IP
BVLC Function:	1-octet	X'05'	Register-Foreign-Device
BVLC Length:	2-octets	X'0006'	Length, in octets, of the BVLL message
Time-to-Live	2-octets	T	Time-to-Live T, in seconds

The Time-to-Live value is the number of seconds within which a foreign device must re-register with a BBMD or risk having its entry purged from the BBMD's FDT. This value will be sent most significant octet first. See J.5.2.2.

J.2.7 Read-Foreign-Device-Table: Purpose

The message provides a mechanism for retrieving the contents of a BBMD's FDT.

J.2.7.1 Read-Foreign-Device-Table: Format

The Read-Foreign-Device-Table message consists of three fields:

BVLC Type:	1-octet	X'81'	BVLL for BACnet/IP
BVLC Function:	1-octet	X'06'	Read-Foreign-Device-Table
BVLC Length:	2-octets	X'0004'	Length, in octets, of the BVLL message

J.2.8 Read-Foreign-Device-Table-Ack: Purpose

This message returns the current contents of a BBMD's FDT to the requester. An empty FDT shall be signified by a list of length zero.

J.2.8.1 Read-Foreign-Device-Table-Ack: Format

The Read-Foreign-Device-Table-Ack message consists of four fields:

BVLC Type:	1-octet	X'81'	BVLL for BACnet/IP
BVLC Function:	1-octet	X'07'	Read-Foreign-Device-Table-Ack
BVLC Length:	2-octets	L	Length L, in octets, of the BVLL message
List of FDT Entries:	N*10-octets		

N indicates the number of entries in the FDT whose contents are being returned. Each returned entry consists of the 6-octet B/IP address of the registrant; the 2-octet Time-to-Live value supplied at the time of registration; and a 2-octet value representing the number of seconds remaining before the BBMD will purge the registrant's FDT entry if no re-registration occurs.

J.2.9 Delete-Foreign-Device-Table-Entry: Purpose

This message is used to delete an entry from the Foreign-Device-Table.

J.2.9.1 Delete-Foreign-Device-Table-Entry: Format

The Delete-Foreign-Device-Table-Entry message consists of four fields:

BVLC Type:	1-octet	X'81'	BVLL for BACnet/IP
BVLC Function:	1-octet	X'08'	Delete-Foreign-Device-Table-Entry
BVLC Length:	2-octets	X'000A'	Length, in octets, of the BVLL message
FDT Entry:	6-octets		

The FDT entry is the B/IP address of the table entry to be deleted.

J.2.10 Distribute-Broadcast-To-Network: Purpose

This message provides a mechanism whereby a foreign device may cause a BBMD to broadcast a message on all IP subnets in the BBMD's BDT.

J.2.10.1 Distribute-Broadcast-To-Network: Format

The Distribute-Broadcast-To-Network message consists of four fields:

BVLC Type:	1-octet	X'81'	BVLL for BACnet/IP
BVLC Function:	1-octet	X'09'	Distribute-Broadcast-To-Network
BVLC Length:	2-octets	L	Length L, in octets, of the BVLL message
BACnet NPDU from Originating Device:			Variable length

J.2.11 Original-Unicast-NPDU: Purpose

This message is used to send directed NPDUs to another B/IP device or router.

J.2.11.1 Original-Unicast-NPDU: Format

The Original-Unicast-NPDU message consists of four fields:

BVLC Type:	1-octet	X'81'	BVLL for BACnet/IP
BVLC Function:	1-octet	X'0A'	Original-Unicast-NPDU
BVLC Length:	2-octets	L	Length L, in octets, of the BVLL message
BACnet NPDU:	Variable length		

J.2.12 Original-Broadcast-NPDU: Purpose

This message is used by B/IP devices and routers which are not foreign devices to broadcast NPDUs on a B/IP network.

J.2.12.1 Original-Broadcast-NPDU: Format

The Original-Broadcast-NPDU message consists of four fields:

BVLC Type:	1-octet	X'81'	BVLL for BACnet/IP
BVLC Function:	1-octet	X'0B'	Original-Broadcast-NPDU
BVLC Length:	2-octets	L	Length L, in octets, of the BVLL message
BACnet NPDU:	Variable length		

J.3 BACnet/IP Directed Messages

B/IP devices shall communicate directly with each other by using the B/IP address of the recipient. Each NPDU shall be transmitted in a BVLL Original-Unicast-NPDU.

J.4 BACnet/IP Broadcast Messages

This clause defines how BACnet broadcast messages are managed within a B/IP network.

J.4.1 B/IP Broadcast Management, Single IP Subnet

In this case, the B/IP network consists of a single IP subnet. A "local broadcast" shall use the B/IP broadcast address and the NPDU shall be transmitted in a BVLL Original-Broadcast-NPDU message. Because all nodes are on a single IP subnet, such messages will automatically reach all nodes. See Figure J-1.

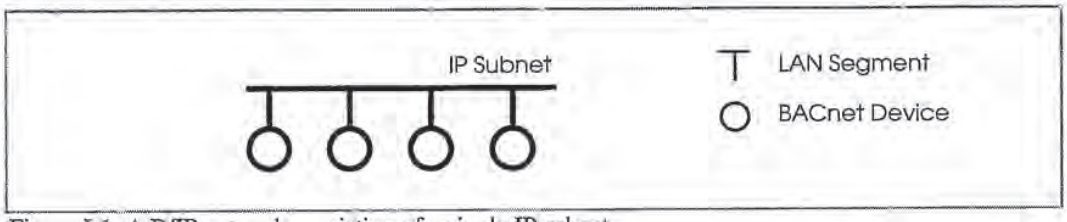

Figure J-1. A B/IP network consisting of a single IP subnet.

J.4.2 B/IP Broadcast Management, Multiple IP Subnets

In this case, the BACnet/IP network consists of two or more IP subnets. A "local broadcast" shall use the B/IP broadcast address, and the NPDU shall be transmitted in a BVLL Original-Broadcast-NPDU message. Because standard IP routers do not forward such broadcasts, an ancillary device is required to perform this function. This device shall be called a BACnet/IP Broadcast Management Device (BBMD). See Figure J-2.

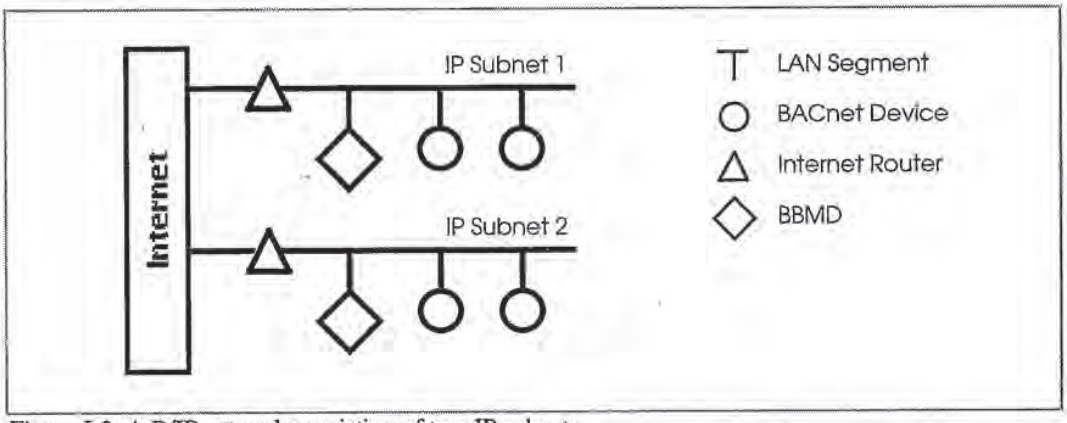

Figure J-2. A B/IP network consisting of two IP subnets.

J.4.3 BBMD Concept

Each IP subnet that is part of a B/IP network comprised of two or more subnets shall have one, and only one, BBMD. Each BBMD shall possess a table called a Broadcast Distribution Table (BDT) which shall be the same in every BBMD in a given B/IP network. If the BBMD has also been designated to register foreign devices as described below, it shall also possess a Foreign Device Table (FDT).

J.4.3.1 Broadcast Distribution

There are two ways that a BBMD may distribute broadcast messages to remote IP subnets. The first is to use IP "directed broadcasts" (also called "one-hop" distribution). This involves sending the message using a B/IP address in which the network portion of the address contains the subnet of the destination IP subnet and the host portion of the address contains all 1's. While this method of distribution is efficient, it requires that the IP router serving the destination subnet be configured to support the passage of such directed broadcasts.

Since not all IP routers are configured to pass directed broadcasts, a BBMD may be configured to send a directed message to the BBMD on the remote subnet ("two-hop" distribution) which then transmits it using the B/IP broadcast address. Since the use of one-hop distribution requires an IP router configuration that may or may not be possible, while the two-hop method is always available, the choice of which method to use in any given case is a local matter.

J.4.3.2 Broadcast Distribution Table Format

The BDT consists of one entry for each BBMD within a B/IP network. Each entry consists of the 6-octet B/IP address of the BBMD serving the IP subnet and a 4-octet broadcast distribution mask. If messages are to be distributed on the remote IP subnet using directed broadcasts, the broadcast distribution mask shall be identical to the subnet mask associated with the subnet, i.e., all 1's in the network portion of the 4-octet IP address field and all 0's in the host portion. If messages are to be distributed on the remote IP subnet by sending the message directly to the remote BBMD, the broadcast distribution mask shall be all 1's. The use of the broadcast distribution mask is described in J.4.5.

J.4.4 BBMD Configuration

The configuration of the BACnet-related capability of a BBMD shall consist of supplying it with a BDT. The table may be supplied by local means or by means of the BVLL Write-Broadcast-Distribution-Table message.

J.4.5 BBMD Operation - Broadcast Distribution

Upon receipt of a BVLL Write-Broadcast-Distribution-Table message, a BBMD shall attempt to create or replace its BDT, depending on whether or not a BDT has previously existed. If the creation or replacement of the BDT is successful, the BBMD shall return a BVLC-Result message to the originating device with a result code of X'0000'. Otherwise, the BBMD shall return a BVLC-Result message to the originating device with a result code of X'0010' indicating that the write attempt has failed.

Upon receipt of a BVLL Read-Broadcast-Distribution-Table message, a BBMD shall load the contents of its BDT into a BVLL Read-Broadcast-Distribution-Table-Ack message and send it to the originating device. If the BBMD is unable to perform the read of its BDT, it shall return a BVLC-Result message to the originating device with a result code of X'0020' indicating that the read attempt has failed.

Upon receipt of a BVLL Original-Broadcast-NPDU message, a BBMD shall construct a BVLL Forwarded-NPDU message and send it to each IP subnet in its BDT with the exception of its own. The B/IP address to which the Forwarded-NPDU message is sent is formed by inverting the broadcast distribution mask in the BDT entry and logically ORing it with the BBMD address of the same entry. This process produces either the directed broadcast address of the remote subnet or the unicast address of the BBMD on that subnet depending on the contents of the broadcast distribution mask. See J.4.3.2.. In addition, the received BACnet NPDU shall be sent directly to each foreign device currently in the BBMD's FDT also using the BVLL Forwarded-NPDU message.

Upon receipt of a BVLL Forwarded-NPDU message, a BBMD shall process it according to whether it was received from a peer BBMD as the result of a directed broadcast or a unicast transmission. A BBMD may ascertain the method by which Forwarded-NPDU messages will arrive by inspecting the broadcast distribution mask field in its own BDT entry since all BDTs are required to be identical. If the message arrived via directed broadcast, it was also received by the other devices on the BBMD's subnet. In this case the BBMD merely retransmits the message directly to each foreign device currently in the BBMD's FDT. If the message arrived via a unicast transmission it has not yet been received by the other devices on the BBMD's subnet. In this case, the message is sent to the devices on the BBMD's subnet using the B/IP broadcast address as well as to each foreign device currently in the BBMD's FDT. A BBMD on a subnet with no other BACnet devices may omit the broadcast using the B/IP broadcast address. The method by which a BBMD determines whether or not other BACnet devices are present is a local matter.

Upon receipt of a BVLL Distribute-Broadcast-To-Network message from a foreign device, the receiving BBMD shall transmit a BVLL Forwarded-NPDU message on its local IP subnet using the local B/IP broadcast address as the destination address. In addition, a Forwarded-NPDU message shall be sent to each entry in its BDT as described above in the case of the receipt of a BVLL Original-Broadcast-NPDU as well as directly to each foreign device currently in the BBMD's FDT except the originating node. If the BBMD is unable to perform the forwarding function, it shall return a BVLC-Result message to the foreign device with a result code of X'0060' indicating that the forwarding attempt was unsuccessful.

J.5 Addition of Foreign B/IP Devices to an Existing B/IP Network

J.5.1 Foreign device definition

A "foreign" device is a BACnet device that has an IP subnet address different from those comprising the BACnet/IP network that the device seeks to join. The foreign device may be a full-time node on the foreign subnet or may be a part-time participant, as would be the case if the device accessed the internet via a SLIP or PPP connection. See Figure J-3.

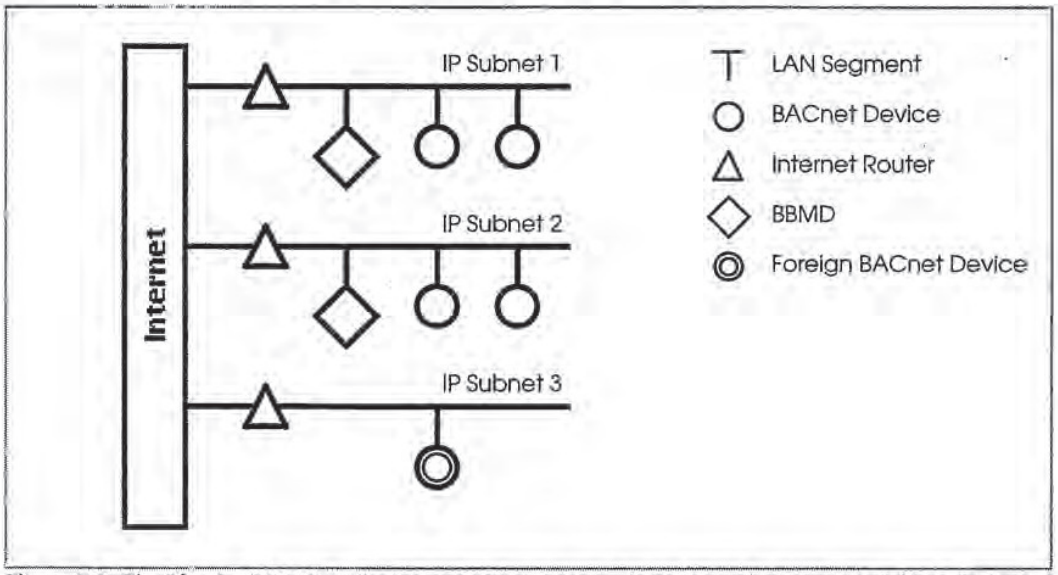

Figure J-3. The "foreign" BACnet device on Subnet 3 can register to receive broadcasts from devices on Subnets 1 and 2 by sending a BVLL Register-Foreign-Device message to a BBMD that supports foreign device registration.

J.5.2 BBMD Operation - Foreign Devices

In order for a foreign device to fully participate in the activities of a B/IP network, the device must register itself with a BBMD serving one of the IP subnets comprising that network. "Full participation" implies the ability to send and receive both directed and broadcast messages. Registration consists of sending a BVLL Register-Foreign-Device message to an appropriate BBMD and receiving a BVLC-Result message containing a result code of X'0000' indicating the successful completion of the registration. Ascertaining the IP address of such a BBMD is a local matter but could involve the use of a domain nameserver or the distribution of a numeric IP address to authorized users. The UDP port X'BAC0' shall be considered the default, but the use of other port values is permitted if required by the local network architecture, e.g., where two B/IP networks share the same physical LAN.

J.5.2.1 Foreign Device Table

Each device that registers as a foreign device shall be placed in an entry in the BBMD's Foreign Device Table (FDT). Each entry shall consist of the 6-octet B/IP address of the registrant; the 2-octet Time-to-Live value supplied at the time of registration; and a 2-octet value representing the number of seconds remaining before the BBMD will purge the registrant's FDT entry if no re-registration occurs. This value will be initialized to the the 2-octet Time-to-Live value supplied at the time of registration.

Two BVLL messages support the maintenance of FDTs and are described in J.5.2.1.1 and J.5.2.1.2.

J.5.2.1.1 Use of the BVLL Read-Foreign-Device-Table Message

Upon receipt of a BVLL Read-Foreign-Device-Table message, a BBMD shall load the contents of its FDT into a BVLL Read-Foreign-Device-Table-Ack message and send it to the originating device. If the BBMD is unable to perform the read of

its FDT, it shall return a BVLC-Result message to the originating device with a result code of X'0040' indicating that the read attempt has failed.

J.5.2.1.2 Use of the BVLL Delete-Foreign-Device-Table-Entry Message

Upon receipt of a BVLL Delete-Foreign-Device-Table-Entry message, a BBMD shall search its foreign device table for an entry corresponding to the B/IP address supplied in the message. If an entry is found, it shall be deleted and the BBMD shall return a BVLC-Result message to the originating device with a result code of X'0000'. Otherwise, the BBMD shall return a BVLC-Result message to the originating device with a result code of X'0050' indicating that the deletion attempt has failed.

J.5.2.2 Use of the BVLL Register-Foreign-Device Message

Upon receipt of a BVLL Register-Foreign-Device message, a BBMD capable of providing foreign device support and having available table entries, shall add an entry to its FDT as described in J.5.2.1 and reply with a BVLC-Result message containing a result code of X'0000' indicating the successful completion of the registration. A BBMD incapable of providing foreign device support shall return a BVLC-Result message containing a result code of X'0030' indicating that the registration has failed.

J.5.2.3 Foreign Device Table Timer Operation

Upon receipt of a BVLL Register-Foreign-Device message, a BBMD shall start a timer with a value equal to the Time-to-Live parameter supplied plus a fixed grace period of 30 seconds. If, within the period during which the timer is active, another BVLL Register-Foreign-Device message from the same device is received, the timer shall be reset and restarted. If the time expires without the receipt of another BVLL Register-Foreign-Device message from the same foreign device, the FDT entry for this device shall be cleared.

Upon receipt of a BVLC-Result message containing a result code of X'0000' indicating the successful completion of the registration, a foreign device shall start a timer with a value equal to the Time-to-Live parameter of the preceding Register-Foreign-Device message. At the expiration of the timer, the foreign device shall re-register with the BBMD by sending a BVLL Register-Foreign-Device message.

J.6 Routing Between B/IP and non-B/IP BACnet Networks

J.6.1 Router Operation

In concept, a router between a B/IP network and a non-B/IP network functions identically to the routers described in Clause 6. See Figure J-4.

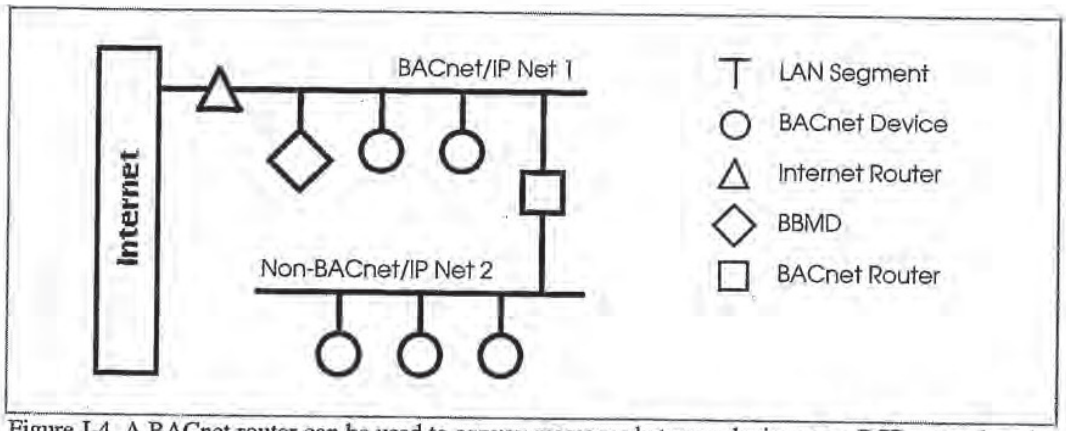

Figure J-4. A BACnet router can be used to convey messages between devices on a B/IP network and non-B/IP network using the procedures in Clause 6.

There are two possible differences. First, on the B/IP side, the B/IP address is used in place of the MAC layer address referred to throughout Clause 6. Second, if B/IP and non-B/IP BACnet devices reside on the same physical LAN, then all traffic is typically sent and received through a single physical port. The collection of B/IP devices would, in such a case, have a network number distinct from the network number of the non-B/IP devices. Such a scenario could easily occur on an Ethernet network where some devices are IP-capable while others are not.

J.7 Routing Between Two B/IP BACnet Networks

Although the foreign registration process provides the ability for remote devices to participate in a particular B/IP network, there may be occasions when it is desirable for two collections of B/IP devices to interoperate more closely. This type of interoperation can only produce results consistent with the assumptions and intent contained in the original BACnet standard if the configuration of the two B/IP networks has been coordinated. For example, it is assumed that Device object identifiers are unique "internetwork wide." If this is not the case, the Who-Is service will produce ambiguous results. Similarly, the Who-Has service may be useless for dynamic configuration applications if multiple instances of objects with identical object identifiers exist.

The BACnet standard also assumes that only a single path exists between devices on different BACnet networks and that this path passes through a BACnet router. The internet's web topology violates this assumption in that, apart from security constraints such as "firewalls", any IP device can communicate directly with any other IP device if it knows the device's IP address.

This clause specifies how B/IP internetworks may be constructed.

J.7.1 B/IP Internetwork Design Considerations

This standard recognizes that BACnet internetworks containing one or more B/IP networks can be configured in a variety of ways, depending on the requirements of an installation. Any of these configurations shall be deemed to conform to this standard provided they employ the techniques specified in this clause.

1) Depending on local traffic conditions and security requirements, all B/IP subnetworks can be configured into a single B/IP network. This case is dealt with in clauses J.1-6.

2) Creating two or more B/IP networks, each with a unique network number, can be useful for limiting the propagation of local broadcast messages and for providing security by confining traffic to a particular geographic or logical area.

3) A single device can be configured to provide all the routing for a B/IP internetwork. See Figure J-5. The advantages include: only a single routing table is required; the possibility of creating multiple paths between B/IP networks is eliminated; the resulting star topology is easy to conceptualize. The disadvantages are: there is a single point of failure; a single device could present a traffic bottleneck under heavy load conditions.

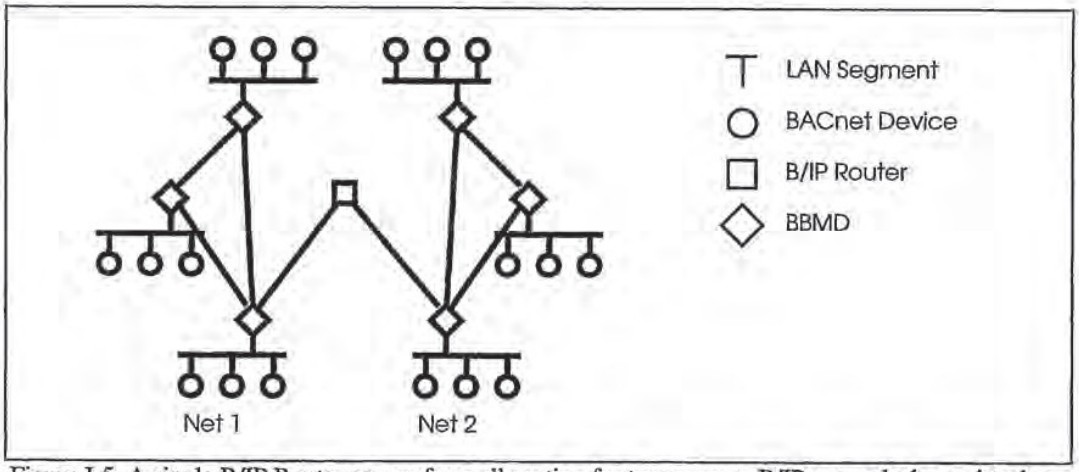

Figure J-5. A single B/IP Router can perform all routing for two or more B/IP networks by registering as a foreign device on each network. It is then "directly connected" to each such network and uses a UDP port unique to that network for the receipt of communications from its individual nodes. See J.7.2. The unique UDP port is required to determine a message's origin for the purpose of appending an SNET to the routed packet. Note that the UDP port associated with the B/IP addresses of the non-router nodes remains in general X'BAC0'.

4) While the functions of BBMDs as specified in J.4 and of BACnet routers as specified in Clause 6 and J.7.2 are entirely distinct, this standard does not preclude the implementation of BBMD functionality and router functionality in a single physical device. See Figure J-6.

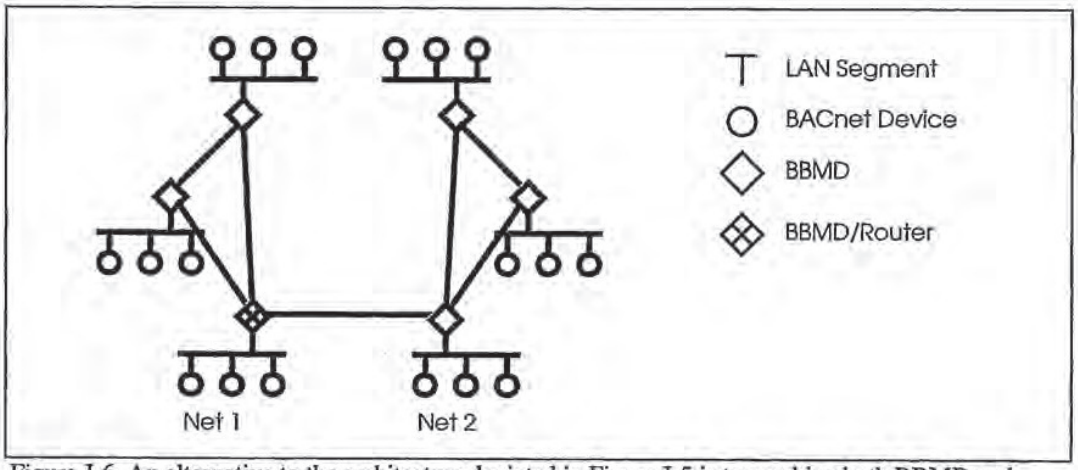

Figure J-6. An alternative to the architecture depicted in Figure J-5 is to combine both BBMD and router functionality in the same physical device as explicitly permitted in J.7.1.

J.7.2 B/IP Routers

B/IP routers adhere to the requirements of Clause 6 with the following differences:

1) The physical ports of Clause 6 routers are replaced by logical ports. Each logical port is identified by the unique B/IP address of the port's connection to the B/IP network served by the router.

2) The term "directly connected network" in Clause 6 implies a physical LAN connection between a LAN segment and a physical router port. In this clause "directly connected network" is extended to mean any B/IP network from which a router can receive local broadcast or IP multicast messages. Such networks are: the B/IP network on which a router resides by virtue of having an IP network number in common with one of the IP subnets comprising the B/IP network; a

B/IP network in which the router participates as a member of an IP multicast group; or a B/IP network in which a router participates by having registered as a foreign device with a BBMD serving that network.

3) Networks that are not directly connected are called "remote networks." Remote networks, whether B/IP or non-B/IP, may be reachable by communicating using B/IP with a router serving the remote network.

J.7.3 B/IP Router Tables

B/IP router tables shall contain the following information for each logical port:

(a) the B/IP address for this port,

(b) if the port is to be used to communicate with nodes on a network directly connected by virtue of the router having an IP network number in common with one of the IP subnets comprising the B/IP network, the BACnet network number of the network, else

if the port is to be used to communicate with nodes on a network directly connected by virtue of the router registering as a foreign device with a BBMD, the BACnet network number of the network served by the BBMD and the B/IP address of the BBMD offering foreign device registration,

(c) a list of network numbers reachable through this port along with the B/IP address of the next router on the path to each network number and the reachability status, as defined in 6.6.1, of each such network.

Because internetworks involving multiple B/IP networks may be more dynamic than traditional BACnet internetworks, implementers of B/IP routers may wish to provide a mechanism whereby specific table entries can be selectively activated and deactivated. The mechanism for accomplishing this is deemed to be a local matter.

J.7.4 B/IP Router Operation

Upon start-up, each B/IP router shall search its routing table for active entries indicating direct connection via foreign registration with a BBMD. The router shall then proceed to register itself with each such network using the procedures specified in J.5. At the conclusion of all such registrations, the router shall follow the procedure of 6.6.2 in that it shall broadcast an I-Am-Router-To-Network message containing the network numbers of each accessible network except the networks reachable via the network on which the broadcast is being made. Note that networks accessed through a given active UDP port that are not directly connected, but are reachable by means of communication with another B/IP router shall, upon router startup, be deemed to be reachable.

Additional router operations with regard to local and remote traffic shall follow the procedures of Clause 6.

J.8 Use of IP Multicast within BACnet/IP

BACnet/IP devices that so desire may alternatively use IP multicasting as a method for distributing BACnet broadcast messages, subject to the constraints imposed in this clause. This is accomplished through the use of an IP class D address which is made up of a single multicast group identifier rather than a combination of network and host IDs. Such devices shall be referred to as B/IP-M devices. The use of IP multicasting also requires that devices comprising a multicast group that reside on more than one IP subnet be served by IP routers capable of supporting IP multicast distribution. (See RFC 1112.) Note that all B/IP-M devices must also be capable of processing unicast messages and must each have a unique unicast IP address. All B/IP devices sharing a common IP multicast address should also share a common BACnet network number.

J.8.1 B/IP Multicast (B/IP-M) concept

For the purposes of BACnet/IP, a B/IP-M group functions logically in the same manner as an IP subnet in the previous clauses. The B/IP multicast group address replaces the B/IP broadcast address for members of the group. The following constraints apply:

1. If the B/IP-M group is part of a BACnet network with B/IP non-multicast devices, there shall be one, and only one, BBMD configured to serve the B/IP-M group devices. A BACnet network comprised solely of B/IP-M devices need not have a BBMD unless foreign devices are to be supported.

2. In order to prevent the receipt of multiple broadcast messages, devices that are in the B/IP-M group and B/IP non-multicast devices may not share the same IP subnet. Note that this does not necessarily preclude them from sharing the same physical LAN if the IP router serving the LAN can support multiple IP subnets.

J.8.2 B/IP-M Use of BVLL Messages

B/IP-M devices shall use the Original-Unicast-NPDU BVLL message for directed messages to any B/IP device within the BACnet/IP network. B/IP-M devices shall use the Original-Broadcast-NPDU BVLL message for the purpose of transmitting BACnet "local" and "global" broadcasts and shall use the B/IP-M group address as the destination IP address.

J.8.3 B/IP-M BBMD Operation

BBMDs function as described in J.4.5 except that the BBMD serving the B/IP-M group must also be a member of the group and that the B/IP-M group address is used analogously to the B/IP broadcast address with respect to the B/IP-M group. It is also required that the BDT entry for each BBMD serving a B/IP-M group shall use a broadcast distribution mask of all 1's to force "two-hop" BBMD-to-BBMD broadcast distribution. This is to prevent the multiple receipt of broadcast messages that would occur if the B/IP-M BBMD were on the same IP subnet as any of the B/IP-M devices themselves and a "directed broadcast" were used. The following paragraphs summarize the relevant operations of BBMDs that serve a B/IP-M group:

Upon receipt of an Original-Broadcast-NPDU via its B/IP-M group address, a BBMD shall forward the message to other entries in its BDT (as well as to any devices in its FDT if the BBMD also supports foreign device registration) as described in J.4.5.

Upon receipt of a Forwarded-NPDU from a peer BBMD, the BBMD shall re-transmit the message using the B/IP-M group address (as well as direct it to any devices in its FDT if the BBMD also supports foreign device registration).

Upon receipt of a BVLL Distribute-Broadcast-To-Network message from a foreign device, the receiving BBMD shall transmit a BVLL Forwarded-NPDU message using the B/IP-M group address as the destination address. In addition, a Forwarded-NPDU message shall be sent to each entry in its BDT as described in J.4.5 as well as directly to each foreign device currently in the BBMD's FDT except the originating node. Error processing is as described in J.4.5.

J.9 Sources for Internet Information

The RFCs referred to in this Annex are available from:

USC/Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292-6695

or online at: WWW.ISI.EDU.

NOTICE

INSTRUCTIONS FOR SUBMITTING A PROPOSED CHANGE TO THIS STANDARD UNDER CONTINUOUS MAINTENANCE

This standard is maintained under continuous maintenance procedures by a Standing Standard Project Committee (SSPC) for which the Standards Committee has established a documented program for regular publication of addenda or revisions, including procedures for timely, documented, consensus action on requests for change to any part of the standard. SSPC consideration will be given to proposed changes according to the following schedule:

Deadline for receipt of proposed changes	SSPC will consider proposed changes at next
February 20	ASHRAE Annual Meeting (normally June)

Proposed changes must be submitted to the Manager of Standards (MOS) in the latest published format available from the MOS. However, the MOS may accept proposed changes in an earlier published format, if the MOS concludes that the differences are immaterial to the proposed changes. If the MOS concludes that the current form must be utilized, the proposer may be given up to 20 additional days to resubmit the proposed changes in the current format.

Specific changes in text or values are required and must be substantiated. The Manager of Standards will return to the submitter any change proposals that do not meet these requirements. Supplemental background documents to support changes submitted may be included.

FORM FOR SUBMITTAL OF PROPOSED CHANGE TO ASHRAE STANDARD UNDER CONTINUOUS MAINTENANCE

(Please type)

1. Submitter: _____
(name—type)

Affiliation: _____

Address: _____ City: _____ State: _____ Zip: _____

Telephone: _____ Fax: _____ E-Mail: _____

I hereby grant the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) the non-exclusive royalty rights, including non-exclusive royalty rights in copyright, in my proposals and I understand that I acquire no rights in publication of this standard in which my proposal in this or other similar analogous form is used. I hereby attest that I have the authority and am empowered to grant this copyright release.

Author's Signature: _____ Date: _____

NOTE: Use a separate form for each comment, completing each section (including Sections 1 and 2) to facilitate processing.

2. Number and Year of Standard:

3. Clause (i.e., Section), Subclause or Paragraph Number, and Page Number:

4. I Propose To: (check one)
[ ] Change to read as shown
[ ] Add new text as shown
[ ] Delete and substitute as shown
[ ] Delete without substitution

(Indicate the proposed change by showing a strikeout line through material to be deleted and underlining material to be added. After showing the text to be changed, insert a horizontal line and state the purpose, reason, and substantiation for the proposed change. Use additional pages if necessary.)

5. Proposed Change:

6. Purpose, Reason, and Substantiation Statements:
(Be brief; provide abstracts of lengthy substantiation, full text should be enclosed for reference on request by project committee members.)

[ ] Check if additional pages are attached. Number of additional pages: _____

NOTE: Use separate form for each comment. Submittals (MS Word 6 preferred) may be attached to e-mail (preferable), submitted on diskettes, uploaded to ASHRAE's ftp site, or submitted in paper form by mail or fax to ASHRAE, Manager of Standards, 1791 Tullie Circle, NE, Atlanta, GA 30329-2305.
E-mail: *change.proposal@ashrae.org*. Ftp server address: *ftp.ashrae.org*, directory: *change.proposal*. Fax: 404-321-5478

ELECTRONIC PREPARATION/SUBMISSION OF FORM FOR PROPOSING CHANGES

An electronic version of each change, which must comply with the instructions in the Notice and the Form, is the preferred form of submittal to ASHRAE Headquarters at the address shown below. The electronic format facilitates both paper-based and computer-based processing. Submittal in paper form is acceptable. The following instructions apply to change proposals submitted in electronic form.

Use the appropriate file format for your word processor and save the file in either Microsoft Word 6.0 (preferred) or higher or WordPerfect 5.1 for DOS format. Please save each change proposal file with a different name (example, prop001.doc, prop002.doc, etc., for Word files—prop001.wpm, prop002.wpm, etc., for WordPerfect files). If supplemental background documents to support changes submitted are included, it is preferred that they also be in electronic form as wordprocessed or scanned documents

Electronic change proposals may be submitted either as files (MS Word 6 preferred) attached to an e-mail (uuencode preferred), files uploaded to an ftp site, or on 3.5" floppy disk. ASHRAE will accept the following as equivalent tot he signature required on the change submittal form to convey non-exclusive copyright:

Files attached to e-mail:	Electronic signature on change submittal form (as a picture; *.tif, or *.wpg), or e-mail address
Files on disk or uploaded to ftp site:	Electronic signature on change submittal form (as a picture; *.tif, or *.wpg), listing of the submitter's e-mail address on the change submittal form, or a letter with submitter's signature accompanying the disk or sent by facsimile (single letter may cover all of proponent's proposed changes).

Submit e-mail, ftp file, or disks containing change proposal files to:
Manager of Standards
ASHRAE
1791 Tullie Circle, NE
Atlanta, GA 30329-2305
E-mail: *change.proposal@ashrae.org*
Ftp server address: *ftp.ashrae.org*, logon to anonymous ftp in directory: *change.proposal.*
(Alternatively, mail paper versions to ASHRAE address or Fax: 404-321-5478.)

The form and instructions for electronic submittal to ASHRAE's ftp site or as attachments to e-mail may be obtained from the Standards section of ASHRAE's Home Page, *http://www.ashrae.org*, or by contacting a Standards Secretary, 1791 Tullie Circle, NE, Atlanta, GA 30329-2305. Phone: 404-636-8400. Fax: 404-321-5478. Email: *standards section@ashrae org*.