Essential Objects and Services

- Device Object
- Analog, Binary, and Multi-State objects
- ReadProperty, WriteProperty
- WhoIs, I-Am
- WhoHas, Ihave
- TimeSynchronization
- ReinitializeDevice
- DeviceCommunicationControl

Device Object

- Device Instance Number
- System_Status
- Object_List
- MS/TP Properties
  - Max_Master
  - Max_Info_Frames

Analog, Binary, Multi-state

- Main property is Present_Value, data type is REAL (Analog), active or inactive (Binary), or an integer index (Multi-state).
- Common properties Object_Identifier, Object_Name, Object_Type, Status_Flags, Event_State
- Additional properties are object specific
- Priority Array for Output objects, optional for Value objects.

Priority Array

Relinquish_Default

ReadProperty

- Required service for every BACnet device
- Used to read the value of a single property in any object.
- The return message includes the object and property details along with the value.
- Specific error messages are defined in Clause 15.

ReadProperty

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

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

WriteProperty

- Optional service for devices
- Used to write a value to a single object property.
- Uses a simple acknowledge - which could be ignored initially when sequencing many writes to improve speed.
- Specific error messages are defined in Clause 15.

WriteProperty

```
WriteProperty-Request ::= SEQUENCE {
    objectIdentifier    [0] BACnetObjectIdentifier,
    propertyIdentifier  [1] BACnetPropertyIdentifier,
    propertyArrayIndex  [2] Unsigned OPTIONAL,
        --used only with array datatype
        -- if omitted with an array the entire
        -- array is referenced
    propertyValue       [3] ABSTRACT-SYNTAX.&Type,
    priority            [4] Unsigned8 (1..16) OPTIONAL
        --used only when property is commandable
}
```

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

WhoIs, I-Am

- Pair used for Device ID to MAC binding
- MAC is derived from source address
- Routing information is derived from NPDU
- I-Am storms, WhoIs ranges
- Most common way for Device binding.

WhoIs, I-Am

```
Who-Is-Request ::= SEQUENCE {
    deviceInstanceRangeLowLimit [0] Unsigned (0..4194303) OPTIONAL,
    deviceInstanceRangeHighLimit [1] Unsigned (0..4194303) OPTIONAL
    -- must be used as a pair, see 16.10
}
```

```
I-Am-Request ::= SEQUENCE {
    iAmDeviceIdentifier BACnetObjectIdentifier,
    maxAPDULengthAccepted Unsigned,
    segmentationSupported BACnetSegmentation,
    vendorID Unsigned
}
```

WhoHas, I-Have

- Pair used for Device ID binding
  - WhoHas Device ID? I-Have Device ID.
  - WhoHas Object ID? I-Have Object ID.
  - WhoHas Object Name? I-Have Object Name.
- MAC is derived from source address
- Routing information is derived from NPDU

WhoHas, I-Have

```
Who-Has-Request ::= SEQUENCE {
    limits SEQUENCE {
        deviceInstanceRangeLowLimit [0] Unsigned (0..4194303),
        deviceInstanceRangeHighLimit [1] Unsigned (0..4194303)
    } OPTIONAL,
    object CHOICE {
        objectIdentifier [2] BACnetObjectIdentifier,
        objectName [3] CharacterString
    }
}
```

```
I-Have-Request ::= SEQUENCE {
    deviceIdentifier BACnetObjectIdentifier,
    objectIdentifier BACnetObjectIdentifier,
    objectName CharacterString
}
```

TimeSynchronization

- Local Time Sync
- UTC Time Sync
- Can be used to update clock on clockless devices.
- Usually designate only one device on a network as a time master.

TimeSynchronization

```
TimeSynchronization-Request ::= SEQUENCE {
    time BACnetDateTime
}

UTCTimeSynchronization-Request ::= SEQUENCE {
    time BACnetDateTime
}



BACnetDateTime ::= SEQUENCE {
    date Date,
    time Time
}
```

ReinitializeDevice

- Used to cold or warm restart a device
- Password optional (in the clear)
- Additional passwords can be used to do alternate activities, such as enable a new ROM or go into Bootloader mode.
- Also used for Backup/Restore procedure

ReinitializeDevice

```
ReinitializeDevice-Request ::= SEQUENCE {
    reinitializedStateOfDevice [0] ENUMERATED {
        coldstart (0),
        warmstart (1),
        startbackup (2),
        endbackup (3),
        startrestore (4),
        endrestore (5),
        abortrestore (6)
    },
    password [1] CharacterString (SIZE (1..20)) OPTIONAL
}
```

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

DeviceCommunicationControl

- Used to disable device communications
- After disabled, device can only respond to
  - DeviceCommunicationControl Enable
  - ReinitializeDevice
  - Power cycle

DeviceCommunicationControl

```
DeviceCommunicationControl-Request ::= SEQUENCE {
        timeDuration [0] Unsigned16 OPTIONAL,
        enable-disable [1] ENUMERATED {
        enable (0),
        disable (1),
        disable-initiation (2)
    },
    password [2] CharacterString (SIZE(1..20)) OPTIONAL
}
```

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