SE8000 Series BACnet Configuration

Configuration Guide for BACnet Protocol
SE8000 Series BACnet Configuration

This covers the SE8300, SER8300, and SE8600 Series Room Controllers and follows the BACnet Protocol Implementation Conformance Statement (PICS) format. The BACnet® communicating controller meets all requirements for designation as an Application Specific Controller (ASC), and the BACnet controller supports the following BACnet Interoperability Building Blocks (BIBBs)

SER8300 and SE8300 PICS

<table>
<thead>
<tr>
<th>Application Service</th>
<th>Designation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Sharing – Read Property - B</td>
<td>DS-RP-B</td>
</tr>
<tr>
<td>Data Sharing – Read Property Multiple - B</td>
<td>DS-RPM-B</td>
</tr>
<tr>
<td>Data Sharing – Write Property - B</td>
<td>DS-WP-B</td>
</tr>
<tr>
<td>Data Sharing - Write Property Multiple Service - B</td>
<td>DS-WPM-B</td>
</tr>
<tr>
<td>Device Management - Time Synchronization - B</td>
<td>DM-TS-B</td>
</tr>
<tr>
<td>Device Management - Device Communication Control - B</td>
<td>DM-DCC-B</td>
</tr>
<tr>
<td>Device Management – Dynamic Device Binding - B</td>
<td>DM-DDB-B</td>
</tr>
<tr>
<td>Device Management – Dynamic Object Binding - B</td>
<td>DM-DOB-B</td>
</tr>
</tbody>
</table>

SE8600 PICS

<table>
<thead>
<tr>
<th>Application Service</th>
<th>Designation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Sharing-COV-B</td>
<td>DS-COV-B</td>
</tr>
<tr>
<td>Data Sharing – Read Property - B</td>
<td>DS-RP-B</td>
</tr>
<tr>
<td>Data Sharing – Read Property Multiple - B</td>
<td>DS-RPM-B</td>
</tr>
<tr>
<td>Data Sharing – Write Property - B</td>
<td>DS-WP-B</td>
</tr>
<tr>
<td>Data Sharing - Write Property Multiple Service - B</td>
<td>DS-WPM-B</td>
</tr>
<tr>
<td>Device Management - Time Synchronization - B</td>
<td>DM-TS-B</td>
</tr>
<tr>
<td>Device Management - Device Communication Control - B</td>
<td>DM-DCC-B</td>
</tr>
<tr>
<td>Device Management – Dynamic Device Binding - B</td>
<td>DM-DDB-B</td>
</tr>
<tr>
<td>Device Management – Dynamic Object Binding - B</td>
<td>DM-DOB-B</td>
</tr>
<tr>
<td>Scheduling-Internal-B</td>
<td>SCHED-I-B</td>
</tr>
</tbody>
</table>

Note: The Room Controllers do not support segmented requests or responses.
Integration and Global Commands for SER8300 & SE8300

The below shows which objects from the controller can be monitored and controlled from the BAS front-end. This is for the SER8300 and SE8300 models only.

Global Command Control Level

- **BAS Front End**
  - (All controllers)
  - **Outdoor Temperature**
  - **Outdoor Temperature & Outdoor Humidity (Enthalpy)**
  - **Outdoor Temperature & HVAC plant current mode**

Specific Devices

- **Specific Devices**
  - (Specific area controllers)
  - **Schedule**
    - **Schedule & Outdoor Temperature**
  - **Restrict user access to controller**
  - **Room temperature for testing and override**

Device Level

- **MSTP Network**
  - **SER8300/SE8300**
  - **Outdoor Temperature (AV10)**
  - **Dehumidification Lockout (MV13)**
  - **Sequence of Operation (MV15)**
  - **System mode (AV16)**
  - **Occupancy Command (MV10)**
  - **Fan Mode (MV17)**
  - **Occupied Heating Setpoint (AV39)**
  - **Unoccupied Heating Setpoint (AV43)**
  - **Occupied Cooling Setpoint (AV40)**
  - **Unoccupied Cooling Setpoint (AV44)**
  - **Main Password (AV56)**
  - **Room Temperature (AV100)**
Integration and Global Commands for SE8600

The below shows which objects from the controller can be monitored and controlled from the BAS front-end. This is for the SE8600 models only.

Global Command Control Level

- Outdoor Temperature
- Outdoor Temperature and HVAC plant current mode
- Schedule
- Restrict user access to controller
- Room temperature for testing and override

Device Level

- Outdoor Temperature (AV101)
- System mode (MV16)
- Occupancy Command (MV10)
- Fan Mode (MV17)
- Occupied Heating Setpoint (AV39)
- Unoccupied Heating Setpoint (AV43)
- Occupied Cooling Setpoint (AV40)
- Unoccupied Cooling Setpoint (AV44)
- Main Password (AV56)
- Room Temperature (AV100)
SE8000 Integration GUI Objects

Objects typically used in a GUI:

- Room Temperature (AV100)
- Occupied and Unoccupied Heat Setpoints (AV39 and AV43)
- Occupied and Unoccupied Cool Setpoints (AV40 and AV44)
- Outdoor Temperature (AV101)
- Supply Temperature (AV102, if available)
- Occupancy Command (MV10)
- System Mode (MV16)
- Auxiliary Output Status (BO98)
- PI Heating Demand (AO21)
- PI Cooling Demand (AO22)
- Window Alarm (BV35)
- Filter Alarm (BV36)
- Service Alarm (BV37)
Configuration Objects

If your BAS allows you to remove objects, Schneider Electric recommends removing all configuration objects once your setup is complete. This prevents unnecessary network polling and traffic.

For wiring, Schneider Electric uses EIA-485 as the physical layer between devices and supervisory controllers. A "Device" represents any product with an active EIA-485 network connection, including Schneider Electric and non-Schneider Electric controllers.

A summary of network specifications are listed below.

Summary Specifications

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Media</td>
<td>Twisted pair 18 AWG, 22 AWG, or 24 AWG (shielded recommended)</td>
</tr>
<tr>
<td>Characteristic Impedance</td>
<td>100-130 ohms</td>
</tr>
<tr>
<td>Distributed capacitance</td>
<td>Less than 100 pF per meter (30 pF per foot)</td>
</tr>
<tr>
<td>Maximum length per segment</td>
<td>4000 feet Note: 18 AWG cable</td>
</tr>
<tr>
<td>Polarity</td>
<td>Polarity sensitive</td>
</tr>
<tr>
<td>Multi-drop</td>
<td>Daisy-chain (no T connections)</td>
</tr>
<tr>
<td>Terminus</td>
<td>1. Devices are installed at both ends of the MS/TP network: 120 Ohms resistor should be installed at each end.</td>
</tr>
<tr>
<td></td>
<td>2. A device is installed at one end of the MS/TP network and a third-party device is installed at the other end. Install an End-Of-Line resistor value that matches the third-party device instruction regarding the End-Of-Line resistors.</td>
</tr>
<tr>
<td></td>
<td>3. Third-party devices are installed at both ends of the MS/TP network. Follow the third-party device instructions regarding the End-Of-Line resistors.</td>
</tr>
<tr>
<td>Network bias resistors</td>
<td>510 ohms per wire (maximum two sets per segment)</td>
</tr>
<tr>
<td>Maximum number of nodes per segment</td>
<td>64 (Schneider Electric devices only)</td>
</tr>
<tr>
<td>Maximum number of nodes per network</td>
<td>128</td>
</tr>
<tr>
<td>Baud rate</td>
<td>9600, 19200, 38400, 57600, 76800, 115200 (Auto detect)</td>
</tr>
</tbody>
</table>
### Wiring Guide

#### Cable Type

Schneider Electric recommends the use of a balanced 22-24 AWG twisted pair with a characteristic impedance of 100-130 ohms and capacitance of 30 pF/ft or lower. A braided shield is also recommended.

#### Impedance

A value based on the inherent conductance, resistance, capacitance, and inductance that represent the impedance of an infinitely long cable. The nominal impedance of the cable should be between 100 ohms and 120 ohms. Using 120 ohms results in a lighter load on the network.

#### Capacitance (pF/ft)

The amount of equivalent capacitive load of the cable (per foot basis). One factor limiting total cable length is the capacitive load. Systems with long lengths benefit from using low capacitance cable (17pF/ft or lower).

#### Network Configuration

EIA-485 networks use a daisy chain configuration. A daisy chain has only one main cable and every network device is connected directly along its path.

Three different network configurations: star, bus, and daisy chain. Only the daisy chain configuration is correct for an EIA-485 network.

Other methods of wiring an EIA-485 network may give unreliable and unpredictable results. There are no troubleshooting methods for these types of networks. Site experimentation may be required with no guarantee of success. As a result, Schneider Electric only supports daisy chain configurations.
Maximum Number of Devices
A maximum of 64 nodes are allowed on a single daisy-chain segment. A node is defined as any device (panel, zone, or repeater) connected to the RS485 network. Terminators do not count as a node.

Add the following to determine the number of nodes on a network:

- One node for each device, including main panels
- One node for each repeater on the chain

The below shows one node for the main SC panel and 4 for the controllers, for a total of 5 nodes. If there are more than 64 devices, install repeaters to extend the network.

**NOTE: End Of Line resistors do not count as nodes.**

EOL Resistor

SC Supervisory Controller

Node 1 Node 2 Node 3 Node 4 Node 5

Example Network with 5 Nodes

Maximum Cable Length
The maximum length of a chain is related to its transmission speed. The longer the chain, the slower the speed. Using proper cable, the maximum length of an EIA-485 daisy chain is 4000 feet. This only works reliably for data rates up to 100,000 bps. Schneider Electric’s maximum data rate is 76,800 bps. If you require a maximum network length of more than 4000 feet, repeaters are required to extend the network.

**EIA-485 Repeaters**
If you have more than 64 devices, or require a maximum network length of more than 4000 feet, repeaters are required to extend the network.
The ideal configuration is to daisy chain the repeaters to the main panel. From each of these repeaters, a separate daisy chain branches off.

Do not install repeaters in series as this may result in network reliability problems. Incorrect use of a repeater in an EIA-485 network as shown below.
End Of Line (EOL) Resistors

MS/TP network must be properly terminated. For daisy chain configurations, you must install an EOL resistor at each end of the daisy chain. Depending on your MS/TP network configuration, the resistance value of the EOL resistor may change. Schneider Electric’s devices are installed at both ends of the MS/TP network. Also, a 120 ohm resistor should be installed at each end.

A Schneider Electric device is installed at one end of the MS/TP network and a 3rd party device is installed at the other end. Make sure you install an End-Of-Line resistor value that matches the 3rd party devices instructions regarding its EOL resistor value. Any 3rd party devices are installed at both ends of the MS/TP network.

Network Adapter

The polarity of the connection to the cable is important. From one module to the other it is important that the same colored wire be connected to “plus” or “+” and the other colored wire be connected to the “minus” or “-” as shown below.

NOTE: The Ref terminal should NEVER be used to wire shields. The 2 shields from each feed of the network connection to a controller should be wired together in the back of the controller and properly protected to prevent any accidental connection to the ground. The joined shield connection should then be grounded at a SINGLE point on the whole segment. More than one ground connection to a shielded wire may induce ground loop noises and affect communication.

BACnet® Communication Wiring (if applicable)

default device name and ID

Default Device Name set to Model number – MAC where:

- MAC is the current MAC address of the device.
- Model number is Schneider Electric part number.

The Device Name upgrades as soon as there is a change to the device MAC address. The Device Name and Device ID properties are writable. Both properties can be renamed from any BACnet® network management tool as long as the tool itself can write to these properties.

SER8300 and SE8300 Models

Default Device ID is set to: 83000 + MAC where MAC is the current MAC address of the device.

The device ID upgrades as soon as there is a change to the device’s MAC. For example, when a SER8300 model with a MAC address of 63 is connected to a network, its default Device ID is 86063.

SE8600UXBXX Models

Default Device ID is set to: 86000 + MAC where MAC is the current MAC address of the device.

The device ID upgrades as soon as there is a change to the device’s MAC. For example, when a SE8600 model with a MAC address of 63 is connected to a network, its default Device ID is 86063.
INTEGRATING SCHNEIDER ELECTRIC DEVICES ON A BACNET MS/TP NETWORK

Before doing any BACnet® integration, make sure you refer to a Schneider Electric PICS document. The PICS document lists all the BACnet® Services and Object types supported by a device. You can find the document at www.schneider-electric.com.

Schneider Electric devices do not support the COV service. COV reporting allows an object to send out notices when its Present-Value property is incremented by a pre-defined value. Since this is not supported at Schneider Electric, special attention should be given to the polling time settings at the Supervisory Controller and Workstation level when using a graphic interface or an application program to read or write to a Schneider Electric object.

Graphical Interfaces

A graphic interface might poll all data linked to the graphic page on a COV basis. If the third-party device does not support COV, the graphical interface relies on a pre-configured polling interval, which is usually in hundredths of milliseconds. Any device containing a monitored object could be subject to network traffic congestion if such a polling interval is used. Schneider Electric strongly recommends a polling interval of 5 seconds (minimum) for any graphical interface. This becomes even more critical in graphics where a single representation might poll many devices. If the proper poll rate is not respected, devices may be reported offline by certain front-ends by saturating the traffic handling capacity of BACnet® MS/TP without COV subscription.

Free Programmed Object or Loops

Read and write MS/TP data on an “If Once” basis or a “Do Every Loop” basis instead of reading or writing to a third-party device’s object directly in the program. Otherwise, any read or write request occurs at the Supervisory Controller’s program scan rate, which may be in hundredths of milliseconds. This can bog down a network as single commands can be sent to all ASC devices down the MS/TP trunks every hundredths of milliseconds.

Programs writing to the devices should have a structure similar to the following:

```
If Once Schedule = On then
  MV10 = Occupied
End If

If Once Schedule = Off Then
  MV10 = Unoccupied
End If
```

OR

```
Do Every 5min
  If Schedule = On Then
    MV10 = Occupied
  Else
    MV10 = Unoccupied
  End If
End Do
```

Retries and Timeouts

In BACnet® integration, you should note the device object of the Supervisory Controller and Operator’s Workstation. This object contains the two following required properties:

1) Retry Timeout

2) Number of APDU Retries

The Retry Timeout specifies the time between re-transmissions if the acknowledgement is not received. Increasing this value may help if you are experiencing problems with controllers dropping off line.

The Number of APDU Retries specifies the number of times unsuccessful transmissions are repeated. If the receiving controller has not received the transmission successfully after this many attempts, no further attempts will be made.

For example, if one of the controllers does not reply to a Supervisory Controller (SC) request, and the SC’s Retry Timeout is set to 2000 milliseconds and the Number of APDU Retries is set to 1 (SC level), the SC sends one other request 2 seconds later. If the MS/TP device does not reply, it is considered off line by the workstation.

Having a Retry Timeout value of 10450 milliseconds and a Number of APDU Retries property set to 3 at the SC level may prevent the device from dropping off line. These properties should also be changed at the workstation level since the workstation issues requests to any MS/TP devices when the graphics are used.
Tips and Things to Know

- Each controller is delivered from the factory with the default MAC address set at 254. At this value, the BACnet® communication is not active and the device does not participate in the token pass. To enable the BACnet® communication, set the local MAC address configuration property of the controller to any valid value from 0 to 127.
- After the initial configuration of your device and if your BAS allows you to remove objects, you should remove all configuration objects to prevent unnecessary polling of unused objects and to help speed up the network.
- In default mode of operation, the device automatically matches its baud rate to the baud rate of the network. Automatic baud rate detection occurs when the MS-TP communication port is initialized (on power up). If the network speed is changed, the device keeps listening at the previously detected speed for 10 minutes before resuming auto-baud. Re-powering the devices forces the auto-baud.
- If the device goes off line, the Room Temperature, Outdoor Temperature, and Occupancy bound parameters are released.
- The BACnet® Data Link layer has two key parameters, the device object name and the device object ID. The device object name must be unique from any other BACnet® device object name on the BACnet® network (not just the MS-TP sub-network). The device object ID must be unique from any other BACnet® device object ID on the entire BACnet® network (not just the MS-TP sub-network).
- Time synchronization can be made through a network even if the controller does not support the full date. Therefore, the device cannot claim conformance to the DeviceManagement – TimeSynchronization - B (DM-TS-B) service. The device object does not have the Local_Time or Local_Date properties.
- Device Name and Device ID properties are writable in Schneider Electric device objects. Both properties can be renamed from any BACnet® network management tool as long as the tool itself gives access to write to these properties.

Troubleshooting

<table>
<thead>
<tr>
<th>Error / Fault</th>
<th>Possible Cause</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller does not come online</td>
<td>Two or more controllers have the same MAC address.</td>
<td>Modify each duplicate address to a unique number.</td>
</tr>
<tr>
<td></td>
<td>The MS-TP network has too many devices.</td>
<td>Do not exceed the maximum number of devices and maximum length allowed by the EIA-485 specifications.</td>
</tr>
<tr>
<td></td>
<td>Too many devices were installed without any repeaters.</td>
<td>Repeaters must be installed</td>
</tr>
<tr>
<td></td>
<td>The MS-TP cable runs are broken</td>
<td>Locate the break and correct the wiring.</td>
</tr>
<tr>
<td></td>
<td>MS-TP connections at the module are reversed</td>
<td>Respect polarity of the wires on a MS-TP network.</td>
</tr>
<tr>
<td></td>
<td>The controller does not have power</td>
<td>Apply power to the controller</td>
</tr>
</tbody>
</table>

Temperature  
www.schneider-electric.com/buildings December 2015
Writing and Binding Behavior

Refer to the proper section as per the table below for BACnet write command behavior for specific objects and functionality.

- **Green =** proper write binding method
- **Orange =** correct write binding method with behavior changes on the user HMI
- **Red =** application locking write binding method

<table>
<thead>
<tr>
<th>Object type</th>
<th>Relinquish Default</th>
<th>Priority Array 4-16</th>
<th>Priority Array 1-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI’s, BI’s &amp; MI’s</td>
<td>Note A)</td>
<td>Note A)</td>
<td>Note A)</td>
</tr>
<tr>
<td>Configuration Properties</td>
<td>Note B)</td>
<td>Note C)</td>
<td>Note C1)</td>
</tr>
<tr>
<td>User HMI objects</td>
<td>Note B)</td>
<td>Note C)</td>
<td>Note C1)</td>
</tr>
<tr>
<td>Status objects</td>
<td>Note D)</td>
<td>Note E)</td>
<td>Note E1)</td>
</tr>
<tr>
<td>Physical hardware output objects</td>
<td>Note D)</td>
<td>Note F)</td>
<td>Note F1)</td>
</tr>
</tbody>
</table>

**Note A)** AI’s, BI’s & MI’s.

Object examples in this category: AI2 / Light Sensor Level, BI29 / UI16 Status, MI33 / Effective Occupancy.

All input objects are read only and cannot be written to independently of the priority array used. These types of points are typically used for statuses and external logic functions.

**Note B)** Configuration Properties and User HMI objects.

Object examples in the configuration property category: AV57 / User Password, BV6 / Force High Level Backlight, MV2 / HMI color.

Object examples in the User HMI category: AV40 / Occupied Cool Setpoint, MV16 / System Mode, MV17 / Fan Mode.

Writing and binding to the relinquish default property is the preferred method to use when setting up network logic using the configuration properties and user HMI objects.

When writing and binding to the relinquish default, the controller will store and archive the new written present value in flash over the network. As soon as the new present value is received, the controller will use this new present value in all its internal control logics and functions.

When writing and binding to the relinquish default, the controller internal control logics and functions are **NOT** by-passed, and the controller will still operate normally.

When writing and binding to the relinquish default, a user can still change user HMI values and an installer can still change configuration properties. In this case, the present value used by the controller internal control logics and functions is the last one received. Either the network present value or a value changed locally at the controller HMI.

**Note C)** Configuration Properties and User HMI objects.

Writing and binding to the priority arrays 4 to 16 property is a method to use IF the required intent is to lock the local HMI and prevent local adjustments made by the user HMI values or the configuration property values. If preventing the local user from accessing or tampering with the local HMI is the goal there are other simpler and more elegant ways to do so:
Proper selection of the user experience of the local HMI. AV2 / User HMI can tailor the user screen to properly present to the user the only local adjustments allowed. In total 12 different HMI user screen options are available. Locking the object by writing to array 4 to 16 is not required. AV56 / Main Password will prevent unauthorized installers from tampering with the configuration properties. Locking the object by writing to array 4 to 16 is not required. AV57 / User Password will prevent unauthorized users from tampering with the user HMI value. Locking the object by writing to array 4 to 16 is not required.

When writing and binding to the priority array 4 to 16, the control will **NOT** store and archive to flash memory and will simply use it in RAM. As soon as the new present value is received in priority arrays 4 to 16, the controller will use this new present value in all its internal control logics and functions.

When writing and binding to the priority array 4 to 16, the controller internal control logics and functions **ARE** bypassed, and the controller will **NOT** operate with its own present values, but will be forced to use the last write commands received on priority arrays 4 to 16 for its built-in internal control logics and functions. **HOWEVER**, since these present values are only stored in RAM and not in flash, if a power reset occurs, the override type function will be lost and the controller will start to operate using the relinquish default value. If the override type function is required, a new write command to priority arrays 4 to 16 is needed.

The controller fully supports native and BACnet compliant priority array 1 to 16 write commands. This simply means that a write value at level 4 has a higher authority than a write at level 9.

When writing and binding to priority array 4 to 16, a user **cannot** change user HMI values and an installer **cannot** change configuration properties. In this case, the present value used by the controller internal control logics and functions is the last write command received at priority array 4 to 16.

You can easily identify an overridden point at the controller in either configuration view, the setpoint view, the service view or the test outputs view. The point will be highlighted in **Red** indicating a write command to priority array 4 to 16.

The **ONLY** way to release the override due to the use of writing to priority array 4 to 16 is to send a write NULL command to the proper priority array currently locking the controller.

**Note C1)** Configuration Properties and User HMI objects.

The behavior is similar to the behavior described in NOTE C with the following exceptions and changes

When writing and binding to the priority array 1 to 3, the controller **WILL** store and archive to flash memory. As soon as the new present value is received in priority array 1 to 3, the controller will use this new present value in all its internal control logics and functions.

When writing and binding to the priority array 1 to 3, the controller internal control logics and functions **ARE** bypassed the same way they are when writing to priority array 4 to 16. The controller will **NOT** operate with its own present values, but will be forced to use the last write commands received on priority array 1 to 3 for its built-in internal control logics and functions. **HOWEVER**, since writing to priority array 1 to 3 will store the value in flash memory, if a power reset occurs, the override type function **will be maintained** and the controller will start to operate right away using the last value written to priority array 1 to 3.
The **ONLY** way to release the override due to the use of writing to priority array 1 to 3 is to send a **write NULL** command to the proper priority array currently locking the controller. Only then will the controller start using the relinquish default value and allow the user or installer to change values.

**Note D)** Status objects and Physical hardware output objects.

No override functions are effective when writing or binding to the relinquish default property of status objects and physical hardware output objects. When writing to the relinquish default property, the internal program is not bypassed and still has priority over the write command on the relinquish default property.

The internal program constantly writes internally to the same location used by the relinquish default property hence a single BACnet write at relinquish default has no effect on status objects and physical hardware output objects.

**Note E)** Status objects.

Object examples in this category: BV36 / Filter Alarm, AV21 / PI Heating Demand. All status objects are writable.

Writing and binding to the priority array 4 to 16 property is a method to use **IF** the requirement is to lock the controller program and prevent statuses from being flagged based on the internal operation of the controller application programming.

When writing and binding to the priority array 4 to 16, the control will **NOT** store and archive to flash memory and will simply use it in RAM. As soon as the new present value is received in priority array 4 to 16, the controller will use this new present value in all its internal control logics and functions.

When writing and binding to the priority array 4 to 16, the controller internal control logics and functions **ARE** bypassed, and the controller will **NOT** operate with its own present values, but will be forced to use the last write commands received on priority array 4 to 16 for its built-in internal control logics and functions. **HOWEVER**, since these present values are only stored in RAM and not in flash, if a power reset occurs, the override type function will be lost and the controller will start to operate using the relinquish default value. **IF** the override type function is required, a new write command to priority array 4 to 16 is needed.

The controller fully supports native and BACnet compliant priority array 1 to 16 write commands. This simply means that a write value at level 4 has a higher authority than a write at level 9.

When writing and binding to priority array 4 to 16, you locally identify if an override is effective on a status object using the service view. The point will be highlighted in **RED** indicating a write command to priority array 4 to 16. In this case, the present value used by the controller internal control logics is by-passed and the value displayed and used is the last write command received at priority array 4-16.

The **ONLY** way to release the override due to the use of writing to priority array 4 to 16 is to send a **write NULL** command to the proper priority array currently locking the controller.
**Note E1)** Status objects.

Object examples in this category: BV36 / Filter Alarm, AV21 / PI Heating Demand. All status objects are writable.

The behavior is similar to the behavior described in NOTE E with the following exceptions and changes.

When writing and binding to the priority array 1 to 3, the controller will store and archive to flash memory. As soon as the new present value is received in priority array 1 to 3, the controller will use this new present value in all its internal control logics and functions.

When writing and binding to the priority array 1 to 3, the controller internal control logics and functions are bypassed the same way they are when writing to priority array 4 to 16. The controller will NOT operate with its own present values, but will be forced to use the last write commands received on priority array 1 to 3 for its built-in internal control logics and functions. **HOWEVER**, since writing to priority array 1 to 3 will store the value in flash memory, if a power reset occurs, the override type function will be maintained and the controller will start to operate right away using the last value written to priority array 1 to 3.

The **ONLY** way to release the override due to the use of writing to priority array 1 to 3 is to send a write NULL command to the proper priority array currently locking the controller. Only then will the controller start using the relinquish default value and allow the user or installer to change values.

**Note F)** Physical hardware output objects.

Object examples in this category: BO95 / BO4 High Speed fan, AO123 / UO11 Analog Status. All physical hardware output objects are writable. **CAUTION NEEDS TO BE EXERCISED** since bypassing the internal control functions of the controller **CAN RESULT** in damage to the equipment.

Writing and binding to the priority array 4 to 16 property is a method to use **IF** the requirement is to lock the controller program and prevent physical hardware output objects from being powered based on the internal operation of the controller application programming.

When writing and binding to the priority array 4 to 16, the control will **NOT** store and archive to flash memory and will simply use it in RAM. As soon as the new present value is received in priority array 4 to 16, the controller will use this new present value in all its internal control logics and functions.

When writing and binding to the priority array 4 to 16, the controller internal control logics and functions **ARE** bypassed, and the controller will **NOT** operate with its own present values, but will be forced to use the last write commands received on priority array 4 to 16 for its built-in internal control logics and functions. **HOWEVER**, since these present values are only stored in RAM and not in flash, if a power reset occurs, the override type function will be lost and the controller will start to operate using the relinquish default value. If the override type function is required, a new write command to priority array 4 to 16 is needed. The controller fully supports native and BACnet compliant priority array 1 to 16 write commands. This simply means that a write value at level 4 has a higher authority than a write at level 9.

When writing and binding to priority array 4 to 16, you locally identify if an override is effective on a status object using the test output view or the service view. The point will be highlighted in **RED** indicating a write command to priority array 4 to 16. In this case, the present value used by the controller internal control logics is bypassed and the value displayed and used is the last write command received at priority array 4-16.
The **ONLY** way to release the override due to the use of writing to priority array 4 to 16 is to send a `write NULL` command to the proper priority array currently locking the controller.

**Note F1)** Physical hardware output objects.

Object examples in this category: BO95 / BO4 High Speed fan, AO123 / UO11 Analog Status. All physical hardware output objects are writable. **CAUTION NEEDS TO BE EXERCISED** since bypassing the internal control functions of the controller **CAN RESULT** in damage to the equipment.

The behavior is similar to the behavior described in NOTE F with the following exceptions and changes

When writing and binding to the priority array 1 to 3, the controller **WILL** store and archive to flash memory. As soon as the new present value is received in priority array 1 to 3, the controller will use this new present value in all its internal control logics and functions.

When writing and binding to the priority array 1 to 3, the controller internal control logics and functions **ARE** bypassed the same way they are when writing to priority array 4 to 16. The controller will **NOT** operate with its own present values, but will be forced to use the last write commands received on priority array 1 to 3 for its built-in internal control logics and functions. **HOWEVER,** since writing to priority array 1 to 3 will store the value in flash, if a power reset occurs, the override type function will **be maintained** and the controller will start to operate right away using the last value written to priority array 1 to 3.

The **ONLY** way to release the override due to the use of writing to priority array 1 to 3 is to send a `write NULL` command to the proper priority array currently locking the controller. Only then will the controller start using the relinquish default value and allow the user or installer to change values.

### General Notes on BACnet Writing and Binding Behavior

If in doubt as to whether a point is overridden using BACnet priority array 1 to 16 or not, open the controller configuration menu, the service view or the test output view. An overridden point will appear in RED in the tables.

All entries in priority array 1 to 16 MUST be set to “null” if normal operation using the internal control functions and factory application program is to be used.

Reinitializing the controller to factory default directly from the configuration interface at the controller **WILL** release ANY and ALL write entries into priority array 1 to 16 on ALL objects.

### Summary for Integrators

AI’s, BI’s & MI’s are not writable and are typically only used for status

**Configuration properties and user HMIOBJects**

- Write and bind to relinquish default if you want the local interface to still be able to modify these settings
- Write and bind to priority array 4 to 16 if you desire to override the local application but not save that override after a power reset
- Write and bind to priority array 1 to 3 if you desire to override the local application and save that override after a power reset

**Status objects and physical hardware output objects**

- Write and bind to relinquish default has no effect since the internal application program constantly overrides the last network write command value
- Write and bind to priority array 4 to 16 if you desire to override the local application but not save that override after a power reset
- Write and bind to priority array 1 to 3 if you desire to override the local application and save that override after a power reset

**Internal controller program under normal operation**

### BACnet Relinquish Default value

Physical hardware outputs are processed by BACnet Priority Array 4 -16 value

---

### BACnet Relinquish Default network write
Appendix A

Important notice to all Niagara integrators
For all intents and purposes, BO95 / BO 4 High Speed Fan Output is used on all examples below.
Where BO95 is the object BACnet instance and BO 4 is the terminal location on the controller
Using native Niagara Set Command for an object

Typically when BACnet device objects are discovered under a BACnet device in Niagara, objects are proxied under the local server database using standard Niagara device and object management processes.

As such using the simple add object command, Niagara creates local proxy extensions of certain of the object properties including the BACnet object present value.
When using the “set” command under Niagara, the local server issues a BACnet write command to the controller without specifying any specific priority array from 1 to 16. As per the BACnet standard, when a controller receives an unspecified write command, the controller automatically assigns it to priority array level 16.
As such, the controller now has an entry in priority array 16 of an object which effectively “overrides” it and by-passes the local control application program.

The know issue with this is that by default, Niagara does not poll or update any of the priority arrays of an object and it is NOT reflected under the proxied object property sheet.

So although you do see the real present value of the object under present value “Out” extension of the object, the displayed priority array 16 “In16” shows a value of NULL. But in effect, it is not NULL.

Viewing if a BACnet array 1 to 16 is actually being used

The only way to positively confirm if a point has an entry in any of the 16 priority array levels is to use the Niagara BACnet “Config” view. Simply open the view and add the objects as needed.
Now to confirm if there is an entry, simply view the property sheet of the BACnet object.

In this view you will confirm an entry at level 16 or any other priority array level.

If the point was overridden by accident and the normal controller operation is required and desired, simply set the overridden priority array to NULL to resume the controller normal operation under the relinquish default property value.

**Binding to a point using the Fallback**

Caution need to be taken when binding objects to specific server logic for central control and applications. The same cautions that apply to the “set” command apply to the binding of a point using the Niagara “fallback” proxy extension.

Binding a logic block or function to the “fallback” sets the same behavior as using the Niagara “set” command. The local server issues a BACnet write command to the controller **without** specifying any specific priority array from 1 to 16. As per the BACnet standard, when a controller receives an unspecified write command, the controller automatically assigns it to priority array level 16.

As such, the controller now has an entry in priority array 16 of an object which effectively **overrides** it and **by-passes** the local control application program.

If the intent of the central logic block or sequence is simply to reset local values and still allow local users to change certain settings (such as setpoints and system mode for example) then the easiest way is to bind the logic block to the BACnet relinquish default property value as described below.
Adding the relinquish default object property for binding

In the Niagara point manager under the device, click on the plus (+) icon to expand all the options and properties of the desired object.

Locate the relinquish default property value and add it to the local server database of proxied objects.

Make sure it is added as a writable object and that it is enabled.
When binding your logic block simply point to the relinquish default property of an object at the “fallback” value. This will directly issue the write command to the relinquish default property value insuring you are not locking down the application at priority array 16.
Technical Support

For any issues with SmartStruxure Solution or SmartStruxure Lite, contact Schneider Electric Technical Support according to your region.

**Americas**
+1-(978)-975-9508: Andover, MA, USA, 8:30am - 5:00pm (EST)
+1-(800)-830-1274: Carrollton, TX, USA, 8:00am - 5:00pm (CST)
+1-(888)-444-1311: Rockford, IL, USA, 8:00am - 5:00pm (CST)

**Europe**
+44-1628-741-147: London, England, UK, 8:00am - 4:30pm (GMT)
+46-40-38-69-00: Malmö, Sweden, 8:00am - 4:15pm (CET/CEST)

**Asia Pacific**
Contact Technical Support at
[https://ecobuilding.schneider-electric.com/support](https://ecobuilding.schneider-electric.com/support)

Schneider Electric is the global specialist in energy management and automation. With revenues of 25 billion in FY2014, our 170,000 employees serve customers in over 100 countries, helping them to manage their energy and process in ways that are safe, reliable, efficient and sustainable. From the simplest of switches to complex operational systems, our technology, software and services improve the way our customers manage and automate their operations. Our connected technologies will reshape industries, transform cities and enrich lives.

At Schneider Electric, we call this **Life Is On.**