BACnet/IP Driver

Table of Contents

BACnet/IP Driver

Help version 1.110

CONTENTS

Overview

The BACnet/IP Driver provides a reliable way to connect BACnet/IP devices to OPC Client applications, including HMI, SCADA, Historian, MES, ERP, and countless custom applications. It provides connectivity to equipment using the BACnet protocol over Ethernet (which is often referred to as "BACnet" or "Annex J").

BACnet Resources

The official BACnet specification, *ANSI/ASHRAE Standard 135-2012 BACnet A Data Communication Protocol for Building Automation and Control Networks*, describes all aspects of the BACnet protocol. It is recommended that users be familiar with the standard BACnet objects and properties discussed in Clause 12 of the specification. Users should also be familiar with the particulars of BACnet/IP outlined in Annex J of the specification. The specification document is available along with many other useful resources through the American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc. (ASHRAE) or the official BACnet website www.bacnet.org.

Setup

Supported Devices

The BACnet/IP Driver can be used successfully with devices that use the BACnet protocol, are visible on an Ethernet network, and support the objects, properties, and services supported by this driver. *For more*

information, refer to the Protocol Implementation Conformance Statement (PICS) that is available from the hardware vendor. Conformance data for this driver is provided in **Supported Objects and Services**.

Communications Protocol

BACnet/IP (Annex J)
🔘 **Note:** This driver requires Winsock V1.1 or higher.
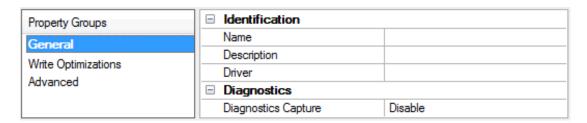
Channel and Device Limits

The maximum number of channels supported by this driver is 128. The maximum number of devices supported by this driver is 128 per channel.

Channel Properties — General

This server supports the use of simultaneous multiple communications drivers. Each protocol or driver used in a server project is called a channel. A server project may consist of many channels with the same communications driver or with unique communications drivers. A channel acts as the basic building block of an OPC link. This group is used to specify general channel properties, such as the identification attributes and operating mode.

Property Groups	⊟ **Identification**	
General	Name	
Write Optimizations	Description	
Advanced	Driver	
	⊟ **Diagnostics**	
	Diagnostics Capture	Disable

Identification

Name: User-defined identity of this channel. In each server project, each channel name must be unique. Although names can be up to 256 characters, some client applications have a limited display window when browsing the OPC server's tag space. The channel name is part of the OPC browser information. The property is required for creating a channel.
🔹 *For information on reserved characters, refer to "How To... Properly Name a Channel, Device, Tag, and Tag Group" in the server help.*

Description: User-defined information about this channel.
🟢 Many of these properties, including Description, have an associated system tag.

Driver: Selected protocol / driver for this channel. This property specifies the device driver that was selected during channel creation. It is a disabled setting in the channel properties. The property is required for creating a channel.
🔘 **Note**: With the server's online full-time operation, these properties can be changed at any time. This includes changing the channel name to prevent clients from registering data with the server. If a client has already acquired an item from the server before the channel name is changed, the items are unaffected. If, after the channel name has been changed, the client application releases the item and attempts to re-acquire using the old channel name, the item is not accepted. With this in mind, changes to the properties should not be made once a large client application has been developed. Utilize the User Manager to prevent operators from changing properties and restrict access rights to server features.

Diagnostics
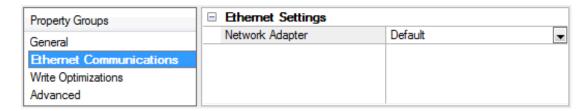
Diagnostics Capture: When enabled, this option makes the channel's diagnostic information available to OPC applications. Because the server's diagnostic features require a minimal amount of overhead processing, it is recommended that they be utilized when needed and disabled when not. The default is disabled.

● **Note:** This property is not available if the driver does not support diagnostics.

❖ *For more information, refer to "Communication Diagnostics" and "Statistics Tags" in the server help.*

Channel Properties — Ethernet Communications

Ethernet Communication can be used to communicate with devices.

Property Groups	Ethernet Settings	
General	Network Adapter	Default
Ethernet Communications		
Write Optimizations		
Advanced		
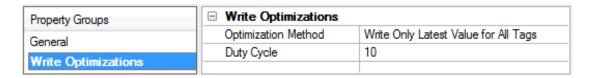
Ethernet Settings

Network Adapter: Specify the network adapter to bind. When left blank or Default is selected, the operating system selects the default adapter.

Channel Properties — Write Optimizations

As with any server, writing data to the device may be the application's most important aspect. The server intends to ensure that the data written from the client application gets to the device on time. Given this goal, the server provides optimization properties that can be used to meet specific needs or improve application responsiveness.

Property Groups	Write Optimizations	
General	Optimization Method	Write Only Latest Value for All Tags
Write Optimizations	Duty Cycle	10

Write Optimizations

Optimization Method: Controls how write data is passed to the underlying communications driver. The options are:

- **Write All Values for All Tags**: This option forces the server to attempt to write every value to the controller. In this mode, the server continues to gather write requests and add them to the server's internal write queue. The server processes the write queue and attempts to empty it by writing data to the device as quickly as possible. This mode ensures that everything written from the client applications is sent to the target device. This mode should be selected if the write operation order or the write item's content must uniquely be seen at the target device.

- **Write Only Latest Value for Non-Boolean Tags**: Many consecutive writes to the same value can accumulate in the write queue due to the time required to actually send the data to the device. If the server updates a write value that has already been placed in the write queue, far fewer writes are needed to reach the same final output value. In this way, no extra writes accumulate in the server's queue. When the user stops moving the slide switch, the value in the device is at the correct value at

virtually the same time. As the mode states, any value that is not a Boolean value is updated in the server's internal write queue and sent to the device at the next possible opportunity. This can greatly improve the application performance.

● **Note**: This option does not attempt to optimize writes to Boolean values. It allows users to optimize the operation of HMI data without causing problems with Boolean operations, such as a momentary push button.

- **Write Only Latest Value for All Tags**: This option takes the theory behind the second optimization mode and applies it to all tags. It is especially useful if the application only needs to send the latest value to the device. This mode optimizes all writes by updating the tags currently in the write queue before they are sent. This is the default mode.

Duty Cycle: is used to control the ratio of write to read operations. The ratio is always based on one read for every one to ten writes. The duty cycle is set to ten by default, meaning that ten writes occur for each read operation. Although the application is performing a large number of continuous writes, it must be ensured that read data is still given time to process. A setting of one results in one read operation for every write operation. If there are no write operations to perform, reads are processed continuously. This allows optimization for applications with continuous writes versus a more balanced back and forth data flow.

● **Note**: It is recommended that the application be characterized for compatibility with the write optimization enhancements before being used in a production environment.

Channel Properties — Advanced

This group is used to specify advanced channel properties. Not all drivers support all properties; so the Advanced group does not appear for those devices.

Property Groups	⊟ **Non-Normalized Float Handling**	
General	Floating-Point Values	Replace with Zero
Write Optimizations	⊟ **Inter-Device Delay**	
Advanced	Inter-Device Delay (ms)	0

Non-Normalized Float Handling: A non-normalized value is defined as Infinity, Not-a-Number (NaN), or as a Denormalized Number. The default is Replace with Zero. Drivers that have native float handling may default to Unmodified. Non-normalized float handling allows users to specify how a driver handles non-normalized IEEE-754 floating point data. Descriptions of the options are as follows:

- **Replace with Zero**: This option allows a driver to replace non-normalized IEEE-754 floating point values with zero before being transferred to clients.
- **Unmodified**: This option allows a driver to transfer IEEE-754 denormalized, normalized, non-number, and infinity values to clients without any conversion or changes.

● **Note:** This property is not available if the driver does not support floating point values or if it only supports the option that is displayed. According to the channel's float normalization setting, only real-time driver tags (such as values and arrays) are subject to float normalization. For example, EFM data is not affected by this setting.

■ For more information on the floating point values, refer to "How To ... Work with Non-Normalized Floating Point Values" in the server help.

Inter-Device Delay: Specify the amount of time the communications channel waits to send new requests to the next device after data is received from the current device on the same channel. Zero (0) disables the delay.

Note: This property is not available for all drivers, models, and dependent settings.

Channel Properties — Protocol Settings

There are three property groups under channel protocol settings. Click on the group of interest for property definitions:

Advanced Settings
Network Settings
Foreign Device

Channel Protocol Properties — Advanced Settings

COV Notifications: This parameter determines if COV notifications from a BACnet device on a different BACnet network require a source address. The choices are **Require NPDU** and **Allow Empty NPDU** (address).

Note: Allowing empty NPDUs may decrease performance on BACnet networks with many COV notifications and/or broadcast requests.

Channel Protocol Settings — Network Settings

The network settings are common to all devices on a channel.

UDP Port: Specifies the local UDP port, as a decimal value, that the driver binds for all communications on the channel. It is also the remote port to which all messages sent to devices on this channel are addressed. The default setting is 47808 (0xBAC0).
Note: Typically, all BACnet/IP devices on an Ethernet network use the same port.

Local Network Number: Specifies the local BACnet/IP network number on which the driver is located. It should be set to the same network number as the local devices. The local network number may range from 1 to 65534. The default setting is 1.

Local Device Instance: Specifies the local BACnet/IP device instance number. Each device on a BACnet internetwork is uniquely identified by its network number and device object instance. This local device number is returned in an "I-Am" service response to a "Who-Is" service request. The valid range is 0 to 4194302. The default setting is 0.
Note: Do not duplicate numbers across the network.

Channel Protocol Settings — Foreign Device

A foreign device is a BACnet/IP device (or software application) that resides on an IP subnet that is not part of a BACnet/IP network. BACnet/IP subnets are considered part of a larger BACnet/IP network if both directed and broadcasted messages can be forwarded to and from the other subnets by IP routers and BACnet Broadcast Management Devices (BBMD).

A foreign device may need to take special measures to discover devices on a BACnet network. For example, if the broadcast Who-Is/I-Am messages normally used for discovery will not be forwarded to and from the foreign device's subnet, the foreign device must work directly with a BBMD on the remote network to discover devices. The foreign device will send broadcast Who-Is messages for the BBMD to then broadcast throughout its BACnet network. I-Am messages broadcasted on the BACnet network are forwarded back to the foreign device if it has registered with the BBMD.

A channel using this driver becomes a foreign device if the selected network interface is not connected to a BACnet/IP subnet. The settings on the channel's Foreign Device must be set to permit the discovery of devices on (or accessible from) a remote BACnet/IP network.

● **Note**: If none of the devices configured on the channel use the driver's Device Discovery feature, ignore the Foreign Device settings.

● *See Also*: *Discovery*, *Configuring Multiple Channels*
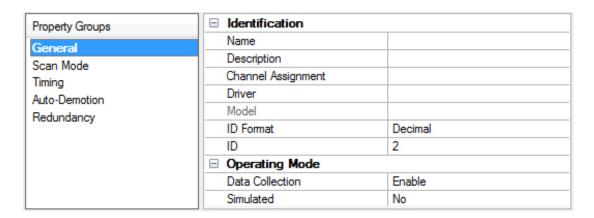
Register as Foreign Device: Enable this property to allow foreign device discovery functionality. Enable if the channel is not connected to a BACnet/IP subnet.

IP Address of Remote BBMD: Specify the IP address of the remote BBMD the driver uses during device discovery.

Registration Time to Live (s): Specify the length of time, in seconds, the BBMD should forward broadcast messages to the driver. The driver only requires an active registration during device discovery and automatically renews the registration as needed. Specify a time long enough for the driver to discover all configured devices (to eliminate the need for renewals and optimize startup performance). A few seconds is generally sufficient unless import on startup is enabled or discovery timeouts are anticipated. Very long times should be avoided to reduce the load on the driver and BBMD after discovery is complete. The valid range is 10 to 3600 seconds. The default setting is 60.

● *For more information, refer to Connection Timeout under **Device Setup**.*

Device Properties — General

Property Groups	☐ **Identification**	
General	Name	
Scan Mode	Description	
Timing	Channel Assignment	
Auto-Demotion	Driver	
Redundancy	Model	
	ID Format	Decimal
	ID	2
	☐ **Operating Mode**	
	Data Collection	Enable
	Simulated	No

Identification

Name: User-defined identity of this device.

Description: User-defined information about this device.

Channel Assignment: User-defined name of the channel to which this device currently belongs.

Driver: Selected protocol driver for this device.
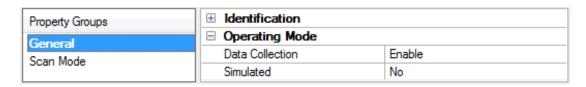
Model: The specific version of the device.

ID: Each device on a BACnet inter-network is uniquely identified by its network number and device object instance. The device ID has the form *<network number>.<device instance>*. For example, to communicate with device 100 on network 1, users would enter "1.100". The network number may range from 1 to 65534 and the device instance may range from 0 to 4194303. The IP address of the device or BACnet gateway / router device is discovered on communications startup by a Who-Is/I-Am exchange and is transparent to the user.

🔹 **Note**: Although each device on a channel must have a unique ID, users may address the same device from separate channels. If a device is configured with the same ID as another device already on that channel, a message warns of the duplication. *For more information, refer to* **Configuring Multiple Channels**. If an invalid ID is written to the "_DeviceId" device System Tag, this message is not posted. Such configuration changes cause communication with that device to fail.

🔹 **See Also**: **Device Properties — Operating Mode**

Operating Mode

Property Groups	⊞ **Identification**	
General	☐ **Operating Mode**	
Scan Mode	Data Collection	Enable
	Simulated	No

Data Collection: This property controls the device's active state. Although device communications are enabled by default, this property can be used to disable a physical device. Communications are not attemp-

ted when a device is disabled. From a client standpoint, the data is marked as invalid and write operations are not accepted. This property can be changed at any time through this property or the device system tags.
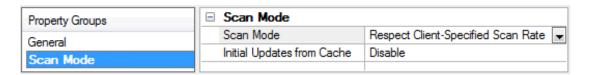
Simulated: This option places the device into Simulation Mode. In this mode, the driver does not attempt to communicate with the physical device, but the server continues to return valid OPC data. Simulated stops physical communications with the device, but allows OPC data to be returned to the OPC client as valid data. While in Simulation Mode, the server treats all device data as reflective: whatever is written to the simulated device is read back and each OPC item is treated individually. The item's memory map is based on the group Update Rate. The data is not saved if the server removes the item (such as when the server is reinitialized). The default is No.

⬤ **Notes:**

1. This System tag (_Simulated) is read only and cannot be written to for runtime protection. The System tag allows this property to be monitored from the client.

2. In Simulation mode, the item's memory map is based on client update rate(s) (Group Update Rate for OPC clients or Scan Rate for native and DDE interfaces). This means that two clients that reference the same item with different update rates return different data.

⬤ Simulation Mode is for test and simulation purposes only. It should never be used in a production environment.

Device Properties — Scan Mode

The Scan Mode specifies the subscribed-client requested scan rate for tags that require device communications. Synchronous and asynchronous device reads and writes are processed as soon as possible; unaffected by the Scan Mode properties.

Property Groups		Scan Mode	
General		Scan Mode	Respect Client-Specified Scan Rate
Scan Mode		Initial Updates from Cache	Disable

Scan Mode: Specifies how tags in the device are scanned for updates sent to subscribing clients. Descriptions of the options are:

- **Respect Client-Specified Scan Rate**: This mode uses the scan rate requested by the client.
- **Request Data No Faster than Scan Rate**: This mode specifies the value set as the maximum scan rate. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
  ⬤ **Note**: When the server has an active client and items for the device and the scan rate value is increased, the changes take effect immediately. When the scan rate value is decreased, the changes do not take effect until all client applications have been disconnected.
- **Request All Data at Scan Rate**: This mode forces tags to be scanned at the specified rate for subscribed clients. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
- **Do Not Scan, Demand Poll Only**: This mode does not periodically poll tags that belong to the device nor perform a read to get an item's initial value once it becomes active. It is the client's responsibility to poll for updates, either by writing to the _DemandPoll tag or by issuing explicit device reads for individual items. *For more information, refer to "Device Demand Poll" in server help.*
- **Respect Tag-Specified Scan Rate**: This mode forces static tags to be scanned at the rate specified in their static configuration tag properties. Dynamic tags are scanned at the client-specified scan rate.

Initial Updates from Cache: When enabled, this option allows the server to provide the first updates for newly activated tag references from stored (cached) data. Cache updates can only be provided when the new item reference shares the same address, scan rate, data type, client access, and scaling properties. A device read is used for the initial update for the first client reference only. The default is disabled; any time a client activates a tag reference the server attempts to read the initial value from the device.

Device Properties — Device Timing

Property Groups	Communication Timeouts	
General	Connect Timeout (s)	3
Scan Mode	Request Timeout (ms)	5000
Timing	Retry Attempts	3
Auto-Demotion	Timing	
	Inter-Request Delay (ms)	0

Connect Timeout: For this driver, a connection is the process of verifying the presence of a BACnet/IP device on the network and successfully reading some basic communications parameters from its device object. This is accomplished by sending a "Who-Is" service request, and then processing the "I-Am" response. Since UDP is used, this does not involve the actual creation of a socket connection. The connection timeout setting is the amount of time that the driver will wait for the I-Am response. If an I-Am message is not received during this time, the driver will assume the local communications settings. Communication with the device may still be possible if the Who-Is/I-Am exchange fails.

● For more information, refer to **APDU**.

Request Timeout: This property specifies the time that the driver will wait for an expected response from the device before retrying or going on to the next request. The valid range is 100 to 9999 milliseconds. The default setting is 1000 milliseconds.
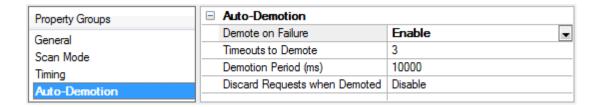
Retry Attempts: This property specifies the number of times that the driver will retry a confirmed request before giving up. The valid range is 1 to 10. The default setting is 3 retries.

Timing Ranges and Defaults

Property	Minimum	Maximum	Default
Connect Timeout	1 s	30 s	3 s
Request Timeout	50 ms	9999999 ms	1000 ms
Retry Attempts	1	10	3

Device Properties — Auto-Demotion

The Auto-Demotion properties can temporarily place a device off-scan in the event that a device is not responding. By placing a non-responsive device offline for a specific time period, the driver can continue to optimize its communications with other devices on the same channel. After the time period has been reached, the driver re-attempts to communicate with the non-responsive device. If the device is responsive, the device is placed on-scan; otherwise, it restarts its off-scan time period.

Property Groups	Auto-Demotion	
General	Demote on Failure	**Enable**
Scan Mode	Timeouts to Demote	3
Timing	Demotion Period (ms)	10000
Auto-Demotion	Discard Requests when Demoted	Disable

Demote on Failure: When enabled, the device is automatically taken off-scan until it is responding again.

 Tip: Determine when a device is off-scan by monitoring its demoted state using the _AutoDemoted system tag.

Timeouts to Demote: Specify how many successive cycles of request timeouts and retries occur before the device is placed off-scan. The valid range is 1 to 30 successive failures. The default is 3.

Demotion Period: Indicate how long the device should be placed off-scan when the timeouts value is reached. During this period, no read requests are sent to the device and all data associated with the read requests are set to bad quality. When this period expires, the driver places the device on-scan and allows for another attempt at communications. The valid range is 100 to 3600000 milliseconds. The default is 10000 milliseconds.

Discard Requests when Demoted: Select whether or not write requests should be attempted during the off-scan period. Disable to always send write requests regardless of the demotion period. Enable to discard writes; the server automatically fails any write request received from a client and does not post a message to the Event Log.

Device Properties — Tag Generation

The automatic tag database generation features make setting up an application a plug-and-play operation. Select communications drivers can be configured to automatically build a list of tags that correspond to device-specific data. These automatically generated tags (which depend on the nature of the supporting driver) can be browsed from the clients.

 Not all devices and drivers support full automatic tag database generation and not all support the same data types. Consult the data types descriptions or the supported data type lists for each driver for specifics.

If the target device supports its own local tag database, the driver reads the device's tag information and uses the data to generate tags within the server. If the device does not natively support named tags, the driver creates a list of tags based on driver-specific information. An example of these two conditions is as follows:

1. If a data acquisition system supports its own local tag database, the communications driver uses the tag names found in the device to build the server's tags.

2. If an Ethernet I/O system supports detection of its own available I/O module types, the communications driver automatically generates tags in the server that are based on the types of I/O modules plugged into the Ethernet I/O rack.

 Note: Automatic tag database generation's mode of operation is completely configurable. *For more information, refer to the property descriptions below.*

On Property Change: If the device supports automatic tag generation when certain properties change, the **On Property Change** option is shown. It is set to **Yes** by default, but it can be set to **No** to control over when tag generation is performed. In this case, the **Create tags** action must be manually invoked to perform tag generation.

On Device Startup: This property specifies when OPC tags are automatically generated. Descriptions of the options are as follows:

- **Do Not Generate on Startup**: This option prevents the driver from adding any OPC tags to the tag space of the server. This is the default setting.
- **Always Generate on Startup**: This option causes the driver to evaluate the device for tag information. It also adds tags to the tag space of the server every time the server is launched.
- **Generate on First Startup**: This option causes the driver to evaluate the target device for tag information the first time the project is run. It also adds any OPC tags to the server tag space as needed.

  🔹 **Note**: When the option to automatically generate OPC tags is selected, any tags that are added to the server's tag space must be saved with the project. Users can configure the project to automatically save from the **Tools | Options** menu.

On Duplicate Tag: When automatic tag database generation is enabled, the server needs to know what to do with the tags that it may have previously added or with tags that have been added or modified after the communications driver since their original creation. This setting controls how the server handles OPC tags that were automatically generated and currently exist in the project. It also prevents automatically generated tags from accumulating in the server.

For example, if a user changes the I/O modules in the rack with the server configured to **Always Generate on Startup**, new tags would be added to the server every time the communications driver detected a new I/O module. If the old tags were not removed, many unused tags could accumulate in the server's tag space. The options are:

- **Delete on Create**: This option deletes any tags that were previously added to the tag space before any new tags are added. This is the default setting.
- **Overwrite as Necessary**: This option instructs the server to only remove the tags that the communications driver is replacing with new tags. Any tags that are not being overwritten remain in the server's tag space.
- **Do not Overwrite**: This option prevents the server from removing any tags that were previously generated or already existed in the server. The communications driver can only add tags that are completely new.
- **Do not Overwrite, Log Error**: This option has the same effect as the prior option, and also posts an error message to the server's Event Log when a tag overwrite would have occurred.

  🔹**Note:** Removing OPC tags affects tags that have been automatically generated by the communications driver as well as any tags that have been added using names that match generated tags.

Users should avoid adding tags to the server using names that may match tags that are automatically generated by the driver.

Parent Group: This property keeps automatically generated tags from mixing with tags that have been entered manually by specifying a group to be used for automatically generated tags. The name of the group can be up to 256 characters. This parent group provides a root branch to which all automatically generated tags are added.

Allow Automatically Generated Subgroups: This property controls whether the server automatically creates subgroups for the automatically generated tags. This is the default setting. If disabled, the server generates the device's tags in a flat list without any grouping. In the server project, the resulting tags are named with the address value. For example, the tag names are not retained during the generation process.
 Note: If, as the server is generating tags, a tag is assigned the same name as an existing tag, the system automatically increments to the next highest number so that the tag name is not duplicated. For example, if the generation process creates a tag named "AI22" that already exists, it creates the tag as "AI23" instead.

Create: Initiates the creation of automatically generated OPC tags. If the device's configuration has been modified, **Create tags** forces the driver to reevaluate the device for possible tag changes. Its ability to be accessed from the System tags allows a client application to initiate tag database creation.
 Note: **Create tags** is disabled if the Configuration edits a project offline.

Device Properties — Protocol Settings

Protocol Settings is a large group of properties, broken down into the following sections:

APDU

Command

COV

COV Shutdown Behavior

Event Notifications

Property Groups	**APDU**	
General	Max. Segments Accepted	Unspecified
Scan Mode	Max. Segments per Window	1
Timing	Max. APDU Length	1476 (fits ISO 8802-3 frame)
Auto-Demotion	Max. Items per Request	16
Tag Generation	**Command**	
Protocol Settings	Command Priority	Priority 8 (Manual Operator)
Tag Import	**COV**	
Discovery	COV Mode	Use Unconfirmed COV
Redundancy	Use SPID of 0 (Select Devices Only)	Disable
	Resubscription Interval (s)	3600
	COV Shutdown Behavior	
	Cancel Subscriptions	Enable
	Await Cancellation ACKs	Disable
	Event Notifications	
	Event Notifications	Disable
	Object Instances Receiving Notifications	

APDU

The Application Protocol Data Unit (APDU) properties affect message segmentation. These limits are imposed by the driver, not the target device. Lower values are automatically used if constrained by the target device. It is generally beneficial to send messages using the largest frame and the fewest segments possible. In most cases, the values are acceptable.

Max. Segments Accepted: Although the driver is not limited in the number of response message segments it can handle, it must specify a limit when making requests. Options include 2, 4, 8, 16, 32, 64, Unlimited, and Unspecified. The default options is Unspecified.

Max. Segments per Window: This property specifies the number of message segments that can be sent before a segment acknowledge message must be returned by the receiving party. The sender proposes a window size, and the receiver determines the actual size (which is no larger than the proposed size). The driver uses this value as the proposed window size for requests and as the actual window size limit for responses from the device. Larger values can increase performance on a reliable network, though smaller settings will allow communications problems to be detected earlier and corrected with fewer segments being resent. The valid range is 1 to 127.

Max. APDU Length: This property specifies the overall length or number of bytes of message segments that the driver will accept. The largest value is generally the best choice. The driver attempts to read the maximum APDU length allowed by the target device on startup and use the smallest of the local or remote limits when sending requests. A smaller value may be needed to accommodate the limitations of hardware between the driver and target device. The driver does not attempt to determine the framing limits of intermediate network devices such as routers and gateways. Options include 50, 128, 206 (fits LonTalk frame), 480 (fits ARCNET frame), 1024, and 1476 (fits ISO 8803-3 frame). The default is 1476 bytes, which is the largest length allowed for BACnet/IP.

Max. Items per Request: This property limits the number of items that can be packed into read property multiple and write property multiple service requests. The actual number of items packed into a request can vary depending on how many items are due for reads or writes at a given time. Generally, the higher the value, the better the performance. For large requests or responses, however, performance gain may be diminished by message segmentation. Unfortunately, there are no general rules for determining the optimum setting. To refine a particular application, experiment with this property. Devices that do not support read property multiple or write property multiple services should be set to 1. The valid range is 1 to 16. The default is 16.

Command

BACnet/IP devices prioritize write requests to certain properties of commandable objects according to a command priority. Once a write to a commandable property has been executed, the sending application acquires command over that property. Write requests from other applications using a lower priority are not be executed until the commanding application relinquishes command over the property. Writes from applications using a higher priority are executed and command is transferred to the higher-priority application.

Control of a supported standard commandable present value property can be relinquished by writing to the PresentValueRel tag. The PresentValueRel tag with present value address and RELINQUISH modifier is created for supported standard commandable objects on automatic tag generation.

This driver supports device-level and object-level prioritization for supported standard commandable objects. Device-level priority is set in the Device Properties. Object-level priority can be set for a supported standard commandable object by writing to its PresentValuePriority tag. The PresentValuePriority tag with

present value address and PRIORITY modifier is created for supported standard commandable objects on automatic tag generation. *For more information on creating special tags, refer to **Address Descriptions**.*

Command Priority

This property specifies the priority level of options for write commands to the device. The priority level can range from 1 (highest) to 16 (lowest). The default setting is 8. The following priority levels have accepted uses as outlined in the BACnet specification:

- **Priority 1**: Manual-Life Safety
- **Priority 2**: Automatic-Life Safety
- **Priority 5**: Critical Equipment Control
- **Priority 6**: Minimum On/Off
- **Priority 8**: Manual Operator

Standard Commandable Properties and Objects

Object	Commandable Property
Analog Output	Present Value
Analog Value	Present Value
Binary Output	Present Value
Binary Value	Present Value
Multi-state Output	Present Value
Multi-state Value	Present Value

● **Note**: Devices may implement additional commandable properties. *For more information, refer to the hardware's PICS statement.*

COV

BACnet allows applications to subscribe to Change of Value (COV) event notification for many properties. When COV notifications are used, the BACnet/IP Driver does not have to continuously poll the device for the current value of these properties. This reduces network traffic and the communications processing load. This driver can be configured to utilize this capability on a per-device level.
● *For more information, refer to **COV Notifications**.*

COV Mode: Indicate how the driver should subscribe to COV notifications for all properties that have implicit and explicit COV support.

- **Use Unconfirmed COV**: When enabled, the driver receives COV notifications from the device without requiring acknowledgment of those notifications. This is the default mode.
- **Use Confirmed COV**: When enabled, the driver receives COV notifications and acknowledges each.
- **Do Not Use COV**: When enabled, all of the device properties are polled even if the COV address modifier is present.

Use SPID of 0 (Select Devices Only): Enable to set the Subscriber Process Identifier (SPID) for all COV items to 0. Disable to use a unique SPID for each subscription. The default setting is disabled. It is only available with Use Unconfirmed COV mode.
● **Tip**: ALC devices consider all subscriptions with an SPID of 0 to be one subscription.
● **Note**: This does not follow the ASHRAE Standard and should only be used by select devices.
● *For information on whether a specific device supports this option, refer to the device manufacturer.*

🔷 *For more information, refer to **Address Descriptions**.*

Resubscription Interval (s): An application can subscribe to COV notifications on a temporary or permanent basis. If 0 is specified, the driver requests permanent subscriptions. In this case, enable Cancel COV Subscriptions to ensure the device can immediately reclaim resources that are no longer needed. Subscribe to temporary subscriptions with a lifetime specified ranging from 1 second to 24 hours. One second prior to the end of the subscription, the driver automatically renews the subscription for active tags.

🔘 **Notes:**

1. If a COV subscription request fails for any reason, the driver polls the device for the associated properties. A message is be placed in the server Event Log indicating when this occurs.

2. Synchronous and asynchronous reads on properties that are configured to rely on COV notifications always read from the cache updated by COV notifications. No direct communication with the device occurs as the result of a synchronous or asynchronous read.

COV Notifications

BACnet provides for unsolicited Change Of Value (COV) reporting of critical properties. The advantage of COV is that the driver does not have to continuously poll the device for these values. Depending on the application, this can result in a significant reduction in network traffic as well as reducing the request processing load on the device and driver.

Implicit vs.Explicit COV
The BACnet specification requires that devices support COV reporting for certain properties. The device may also support COV reporting for other properties, depending on the implementation. Tag addresses to those properties that are required to support COV, are said to have implicit COV support. To take advantage of COV capability of other properties, if implemented, users must add the COV modifier to the tag's address. These tags are said to have explicit COV support. *For more information on address syntax, refer to **Address Descriptions**.*

COV Subscription
Before the driver can receive COV notifications for a particular property, it must first issue a COV subscription request. If the subscription request succeeds, the driver will receive the initial value of the property and COV otifications whenever necessary for as long as the subscription is in effect. The driver will re-subscribe as needed. If a subscription attempt fails, the driver will issue a message to that effect in the server's Event Log. Users may choose to configure the driver to begin polling that property if the subscription request fails. *For configuration details, refer to **COV Settings**.*

COV Mode
COV notifications can be confirmed or unconfirmed. Confirmed notifications require the driver to acknowledge the notification. Unconfirmed notifications are not acknowledged. The driver supports both modes of operations, along with a COV disabled mode where all tags are polled.

COV Watchdog Tags
The OPC quality of tags updated by COV notifications may be GOOD, even if the device has momentarily gone offline. Polled tags, by comparison, can quickly reveal a communications problem when an expected read response fails to arrive. When a poll fails, the driver will flag the device as being in an error state. The driver uses the device error state to set the quality of COV data. Therefore, users interested in monitoring just COV properties should consider polling for one additional property in the device. That polled tag will act as a watchdog for all COV data from that device.

Notes:

1. The BACnet SubscribeCOV service is used for properties with implicit COV support. The BACnet Sub-scribeCOVProperty service is used for all other properties addressed with the COV address modifier.

2. If a COV subscription request fails for any reason, the driver will poll the device for the associated properties. A message is placed in the server's Event Log indicating when this occurs.

COV Shutdown Behavior

BACnet allows applications to subscribe to change of value (COV) event notification for many properties. While this reduces network traffic and the communications processing load, it requires protocols for graceful termination. The following options are available for ending subscriptions.

Cancel Subscriptions: Enable to direct the driver to send messages to the device on driver shutdown to cancel each COV subscriptions. Although this may slightly delay the driver's shutdown, it can be important if the device has limited resources for subscriptions (and if the subscription lifetime is long or permanent). The default is enabled.

Await Cancellation ACKs: Enable to terminate subscriptions one at a time, waiting for the device to reply with an acknowledgment before canceling the next subscription. It also prevents the Runtime from shutting down until all COV subscriptions have been canceled. This option is only available when **Cancel COV sub-scriptions** is enabled. Enabling this property may be helpful for a device that cannot process multiple subscription cancellations at once.
Note: In large projects, this option may appear to stall the Runtime. The Runtime recovers once all subscriptions have been cancelled.

Event Notifications

BACnet allows devices to have objects configured with event reporting to detect and report OffNormal, Fault, or Normal conditions. When an event-initiating object detects the condition, the device sends a confirmed or unconfirmed event notification service request to the configured recipient list. The information provided in these event notifications can be used to update **event-related properties** of the event-initiating intrinsic object or the algorithmically monitored object and the event-initiating event enrollment object. The BACnet/IP Driver does not need to continuously poll the device for the current value of these properties, which reduces network traffic and the communications processing load. This driver can be configured to utilize this capability on a per-object level.

Event Notifications: Enable to notify the driver that some device objects are configured with event reporting, which sends confirmed and/or unconfirmed event notification service requests for condition changes. When enabled, polling no longer occurs on the **event-related properties** of the object instances listed in this property window because these properties are updated by the event notifications. The default setting is Disable.

Object Instances Receiving Notifications: Object instances in this list should be configured with event reporting in the device as one of the following:

- an event-initiating intrinsic object instance
- an event-initiating Event Enrollment object instance
- an algorithmically monitored object instance

Click the property **browse** button to specify the list of unique object instances for which the driver receives event notifications. The browse button launches the Event Notifications dialog that allows **adding** or removing object instances to or from the list property.

 Notes:
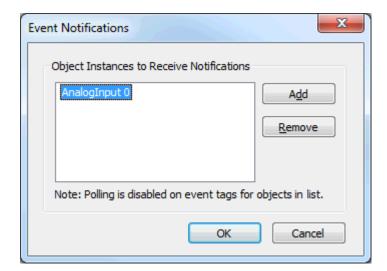
1. Valid object types are those listed under **Supported Objects Types**. The range for the object instance is 0 – 4194302. Click the **...** (Browse) button to **add object instances**.

2. When algorithmic reporting is configured, the EventEnrollment object instance and its Object Property Reference (to prevent polling the PresentValue and StatusFlags properties) must both be in this list.

Caution: When Event Notifications are enabled and an object instance is in the list, polling does not occur for that object's **event-related properties**. To receive updates for those tags after the initial read, event reporting must be properly configured in both the physical device and the BACnet/IP Driver. Confirm the tags are updated when events occur to prove the device is configured to send event notifications for that object instance. If the driver is expected to receive confirmed event notifications, the channel local device number or IP address must be correctly added as a recipient for the event in a notification class of the device, otherwise the device may broadcast those notifications. While the BACnet/IP Driver does respond to confirmed event notifications with a simple ACK, the device may not accept a response to confirmed event notifications that it has broadcast.

 See Also: *Optimizing BACnet Communications*

Event Notifications - Object Instances

This dialog is used to add object instances to the Object Instances Receiving Notifications list. The object type must be among the **Supported Objects Types for Event Notifications** and the instance number falls within the range of 0 - 4194302. The dialog prevents addition of duplicate entries into the Object Instances Receiving Notifications list.
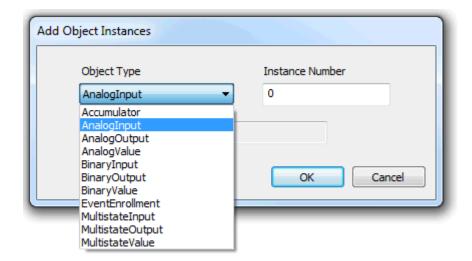
Add Object Instances

The object identifier is comprised of a BACnet object type and an instance number, as defined in the BACnet specification. Add an event-initiating intrinsic object or an algorithmically monitored object and the event-initiating event enrollment object to the Object Instances Receiving Notifications list to prevent polling on the event-related properties of these objects.

Add: Click to launch the Add Object Instances dialog, used to configure an object type and instance to be added to the Object Instances Receiving Notifications list.

Remove: Click to remove the selected object instance from the list.

Object Type: Select the BACnet object type of the object identifier that is configured with intrinsic or algorithmic event reporting or is configured as the object property referenced in an event enrollment object. The default selection is AnalogInput.

Instance Number: Specify the instance of the object identifier that is configured with event reporting. The default instance number is 0. The valid range is 0 – 4194302.

Object Identifier: This read-only property displays the BACnet object identifier comprised of the configured BACnet object type and instance number, as defined in the BACnet specification.

🛢 *See Also*: **Supported Object Types for Event Notifications**.

Event-Related Properties

EventTimeStamps, EventState, StatusFlags, and whatever the *Monitored Value Parameter* is for the object are referred to as the event-related properties. Within an object, the *Monitored Value Parameter* is monitored for conditions that trigger notifications. For most objects, the *Monitored Value Parameter* is PresentValue. However, the Accumulator object *Monitored Value Parameter* is PulseRate.

Intrinsic Reporting

The EventTimeStamps, EventState, StatusFlags, and *Monitored Value Parameter* event-related properties all belong to the event-initiating object reported in the event notification from an intrinsically configured event.

Algorithmic Reporting

The EventState and EventTimeStamps event-related properties belong to the event-initiating Event Enrollment object reported in the event notification from an algorithmically configured event. The StatusFlags and *Monitored Value Parameter* event-related properties belong to the object instance configured as the Event Enrollment object instance Object Property Reference.

Supported Object Types for Event Notifications

0 - Analog-input
1 - Analog-output
2 - Analog-value
3 - Binary-input
4 - Binary-output
5 - Binary-value
9 - Event-enrollment
13 - Multi-state-input
14 - Multi-state-output
19 - Multi-state-value
23 - Accumulator

Device Properties — Tag Import

This driver has the ability to automatically create tags for almost all of the supported device properties. The import (tag generation) can be from a device or from a Cimetrics OPC Server export file. Tag Import can be used to specify the object types that tags are generated in addition to other tag generation options. For information on creating a tag generation from the device, refer to the instructions below.

 The Tag Generation property group controls when the automatic tag generation occurs. For more information, refer to the server help documentation.

Property Groups	Tag Import	
General	Import Method	Device
Scan Mode	Import File	*.csv
Timing	Exclude Optional Properties	Disable
Auto-Demotion	Create Tags as Read/Write if Allowed	Disable
Tag Generation	Use Object Names for Group Names	Disable
Protocol Settings	**Tag Import Objects**	
Tag Import	Enable All	Enable All...
Discovery	Disable All	Disable All...
Redundancy	Accumulator	Disable
	Analog Inputs	Enable
	Analog Outputs	Enable
	Analog Value	Disable
	Averaging	Disable
	Binary Inputs	Enable
	Binary Outputs	Enable
	Binary Values	Disable
	Calendar	Disable
	Command	Disable
	Device	Disable
	Event Enrollment	Disable
	File	Disable
	Group	Disable
	Life Safety Point	Disable
	Life Safety Zone	Disable
	Loop	Disable
	Multistate Input	Disable
	Multistate Output	Disable
	Multistate Value	Disable
	Notification Class	Disable
	Program	Disable
	Schedule	Disable
	Trend Log	Disable

Tag Import

Import Method: Tags can be imported in one of two ways. Select **Device** to import tags from an online device on a network visible to the driver at the time of import. Select **Cimetrics Server CSV File** to import tags exported to a CSV file from the Cimetrics OPC Server. Although export files may contain data for multiple devices, the driver only imports data for this device.

Import File: Specify the path and file name of the comma-separated value (CSV) file to import.
Note: This property is disabled if device is the chosen import method.

Exclude Optional Properties: Generating tags for all the supported properties imported can result in a very large number of tags. To reduce the number of tags that are generated, select Enable to generate tags only for properties required by the BACnet specification. Required properties are those with conformance code R (readable) or W (writable). Tags for non-standard properties and properties with conformance code O (optional) are not generated. Disable to generate tags for all supported properties.

Create Tags as Read/Write if Allowed: Tags are generated with read-only access for properties with a conformance code of R or with Read/Write access if the conformance code is W. The default access for

properties with a conformance code of O depends on the nature of the data. Some BACnet/IP devices allow writes to properties that are described as Read Only in the BACnet specification. The BACnet specification does not specifically forbid this for most properties. All tags may be generated with full Read/Write access to accommodate these non-standard implementations.

📀 *For more information, refer to **Address Descriptions**.*

🔘 **Note**: This property is disabled if Cimetrics Server CSV File is selected as the import method, because the access level is given in the file.

Use Object Names for Group Names: New tag groups are given the name of the corresponding BACnet object. If the object name is not defined or is not unique, the driver assigns a default name to the group.

🔘 **Note**: This property is disabled if Cimetrics Server CSV File is selected as the import method, because the name is not given in the file.

Tag Import Objects

Enable All: Automatically enable all of the object types. It does no harm to enable an object type that does not exist in the device at the time of tag import.

Disable All: Disables all object types.

Device Properties — Discovery

The device ID (set in the General property group in device properties) is sufficient to uniquely identify a device on a BACnet network. This driver requires additional information to establish communication with a BACnet device; such as IP address of the device or router, framing constraints, and the BACnet MAC (Medium Access Control) address of the device. The Discovery property group controls how the driver obtains the necessary BACnet communication parameters.

Property Groups	Device Discovery	
Tag Generation	Discovery Method	Automatic Using Who-Is/I-Am
Protocol Settings	Discovery Scope	Local
Tag Import	Automatic IP Address	255.255.255.255
Discovery		

Discovery Method: Select how the driver obtains the required device properties.

- **Automatic Discovery Using Who-Is/I-Am**: The default selection allows automatic collection of the required communication properties with standard calls.
- **Manual Configuration**: Select this option if the device does not support the I-Am and Who-Is services or if broadcasting messages on the network is undesired.

Discovery Scope: choose how the driver broadcasts Who-Is messages. Four options are available:

- **Local**: Who-Is messages are broadcast over the local Ethernet subnet. Devices on remote Ethernet subnets cannot see these messages. BACnet gateways visible from a local subnet can forward these messages to non-BACnet/IP subnets.
- **Global**: Who-Is messages are broadcast over the entire Ethernet network. Devices on remote Ethernet subnets see these messages unless network routers have been configured to block broadcasts between subnets. In this fairly common scenario, a BBMD must be placed on each Ethernet subnet to forward broadcast BACnet messages.

- **Remote**: Who-Is messages are sent with the global broadcast IP (255.255.255.255), but contain information for BACnet routers and BBMDs to forward them to a single destination network. The destination BACnet network is set in the device ID.
- **Direct**: Who-Is messages are sent directly to a specified IP address. A BACnet gateway at the specified IP address can forward these messages to non-BACnet/IP subnets.

Automatic Discovery Method Options

Automatic IP Address: This property, enabled only when the Discovery Scope is set to Direct, defines the network nodes to which the driver sends messages and from which it accepts responses. This is the IP address of the device if that device is on the local Ethernet network. If the destination device is on a remote network, this must be the IP address of the local router through which communications are conducted.

Manual Configuration Discovery Method Options

Manual IP Address: defines the network nodes to which the driver sends messages and from which it accepts responses. This is the IP address of the device if that device is on the local Ethernet network. If the destination device is on a remote network, this setting must be the IP address of the local router through which communications are conducted.

Remote Data Link Technology: This property controls whether the driver automatically calculates the BACnet MAC for a BACnet/IP device, or if the driver uses a hex string in BACnet MAC. If a device is on a remote subnet that the uses a Data Link Technology that is not BACnet/IP (such as MS/TP, LonTalk, ARCNET); the BACnet MAC must be entered manually and the property should be set to Enable. The default setting is Disable.

> **BACnet MAC**: Although the driver communicates using the BACnet/IP protocol, it is possible to communicate with devices using other Data Link Technologies (such as MS/TP, LonTalk, ARCNET) if the driver is communicating through a gateway. In this case, the BACnet MAC of the underlying device must be entered. The BACnet MAC is a hex string.
>
> **Example**: An MS/TP device is connected to a gateway. The MS/TP has an MAC listed in the web configuration as 10. Since the web configuration displays the value in decimal, the value entered in BACnet MAC should be 0a.
>
> ● **Notes**:
>
> 1. This property is only enabled if the Remote Data Link Technology property is set to Enable.
>
> 2. If the Network Interface selected for the channel is not connected to a BACnet/IP network, configure the channel to operate as a foreign device to discover devices.
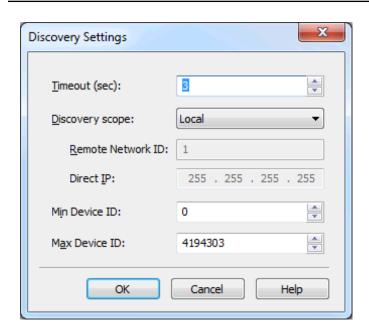
● *For more information, refer to **Foreign Device**.*

Device Discovery

This dialog is used to specify parameters for locating devices on the network. Once devices are found, they may be added to the channel. The maximum number of devices that can be discovered at once is 65535.
● *For more information on device discovery, refer to the server help file.*

Discovery Settings

Timeout: This parameter specifies the length of time that the driver waits for all "I-Am" responses to the initial "Who-Is" discovery request. It is also used to timeout non-responsive devices when requesting device names. The default setting is 3 seconds.

Discovery Scope: This parameter controls how the driver broadcasts "Who-Is" messages. It can also be used to limit the list of devices discovered. Options include Local, Global, Remote, and Direct. The default setting is Local. Descriptions of the options are as follows:

- **Local**: When selected, "Who-Is" messages are broadcast over the local Ethernet subnet. Devices on remote Ethernet subnets cannot see these messages. BACnet gateways visible from local subnets can forward these messages to non-BACnet/IP subnets.
- **Global**: When selected, "Who-Is" messages are broadcast over the entire Ethernet network. Devices on remote Ethernet subnets see these messages unless network routers are configured to block broadcasts between subnets. In this case, a BBMD must be placed on each Ethernet subnet to forward broadcast BACnet messages.
- **Remote**: When selected, "Who-Is" messages are sent with the global broadcast IP 255.255.255.255, but contain information so that BACnet routers and BBMDs forward them to a single destination network. The destination BACnet network is given in the Remote Network ID.
- **Direct**: When selected, "Who-Is" messages are sent directly to the IP address specified in the Direct IP field.

Remote Network ID: This parameter specifies the remote Network ID to be used for remote discovery scope. The default setting is disabled. When enabled, the default value is 1.

Direct IP: This parameter specifies the IP address to be used for direct discovery scope.

Min. Device ID: This parameter specifies the lower range for device discovery. It is used to reduce the number of discovered devices. The valid range is 0 to 4191302. The default setting is 0.
● **Note**: The specified value must be lower than the Max. Device ID.
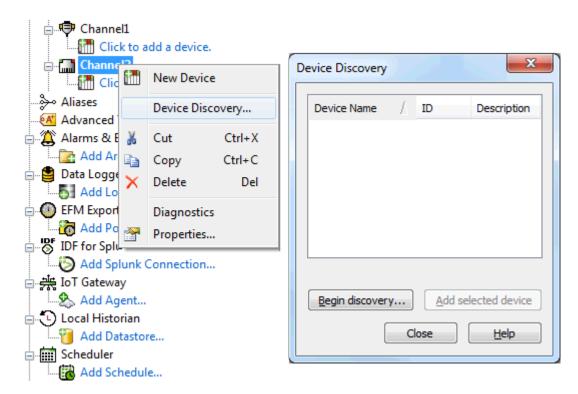
Max. Device ID: This parameter specifies the upper range for device discovery. It is used to reduce the number of discovered devices. The valid range is Min Device ID +1 to 4194303. The default setting is 4194303.

🔘**Note**: The specified value must be higher than the Min. Device ID.
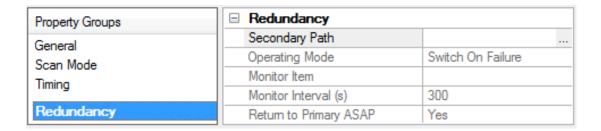
Device Discovery Procedure

Device Discovery is available for drivers that support locating devices on the network. Once devices are found, they may be added to a channel. The maximum number of devices that can be discovered at once is 65535.

1. Select the channel in which devices should be discovered and added.

2. Right click on the channel node and select **Device Discovery...**

3. Click the **Begin discovery...** button to start the discovery process.

4. Specify the discovery properties, which are driver-specific, such as address range, timeout, discovery scope.

5. Click **OK**.

6. Devices discovered populate the dialog with the following information / headings **Name**, **ID**, **Description**.

7. If any discovered device is of interest, select that device and click **Add selected device...**.

8. Click **Close**.

Device Properties — Redundancy

Property Groups		⊟ **Redundancy**	
General		Secondary Path	...
Scan Mode		Operating Mode	Switch On Failure
Timing		Monitor Item	
Redundancy		Monitor Interval (s)	300
		Return to Primary ASAP	Yes

Redundancy is available with the Media-Level Redundancy Plug-In.

🔷 *Consult the website, a sales representative, or the user manual for more information.*

Supported Objects and Services

The following summarizes the portions of the BACnet protocol that are supported by this driver. It should be compared with the hardware's Protocol Implementation Conformance Statement (PICS) that is available from the hardware vendor.

Supported Objects

Supported Services

BACnet Service	BIBB*	Initiate	Execute
Who-Is	DM-DDB-A DM-DDB-B	X	X
I-Am	DM-DDB-A DM-DDB-B	X	X
ReadProperty	DS-RP-A DS-RP-B	X	X
ReadPropertyMultiple	DS-RPM-A DS-RPM-B	X	X
WriteProperty	DS-WP-A	X	
WritePropertyMultiple	DS-WPM-A	X	
SubscribeCOV	DS-COV-A	X	
SubscribeCOVProperty	DS-COVP-A	X	
ConfirmedCOVNotification	DS-COV-A		X
UnconfirmedCOVNotification	DS-COV-A		X
ConfirmedEventNotification Supported Event Types: • Change-of-state • Change-of-value • Command-failure • Out-of-range • Unsigned-range	AE-N-A		X
UnconfirmedEventNotification Supported Event Types: • Change-of-state • Change-of-value • Command-failure • Out-of-range • Unsigned-range	AE-N-A		X

 * **Note**: The BACnet Interoperability Building Block (BIBB) describes the services supported by a BACnet/IP device or application. *For more information, refer to Annex K of the BACnet specification.*

Data Link Layer Support

BACnet/IP (Annex **J**)

Segmentation Support

The BACnet/IP Driver supports both segmented requests and segmented responses. Both requests and responses support window sizes between 1 and 127 bytes.

Character Set Support

ISO 10646 (UTF-8)
IBM/Microsoft DBCS*
ISO 10646 (UCS-2)
ISO 8859-1

All trademarks and registered trademarks are the property of their corporate owners.

Optimizing Communications

Use Multiple Channels
Although this driver has been designed to service read and write requests to multiple devices simultaneously, all pending requests on a channel must be completed before the next set of requests for that channel's devices can be issued. If one device is slow or not responding, it will degrade the performance of all devices on that channel. Each channel operates independently; therefore, it is recommended that users separate devices into several channels for optimum performance. *For more information on the special system requirements for a multi-channel configuration, refer to* **Configuring Multiple Channels**.

Maximize APDU Length
Messages are sent in multiple segments if necessary. To reduce the overhead incurred by message segmentation, use the largest number possible. Before the driver begins reading and writing data to a device, it reads the Max APDU Length of that device. From that point on, the driver uses its maximum APDU length or the device's maximum APDU length, whichever is smallest. Selecting the larger APDU length results in the optimum frame size. However, the driver does not try to see if any network hardware between it and the device (such as BACnet routers and gateways) has more restrictive limits. It may be necessary to reduce the driver's limit to accommodate.

Maximize Segments per Window
The number of message segments that can be sent before the receiving party must return a segment acknowledgment. The sender must wait for this acknowledgment before sending the next series of message segments. Maximizing the segments per window reduces the amount of time consumed waiting for acknowledgments; however, this must be done with the knowledge that communication errors are not detected as quickly and more data must be resent to correct the problem.
● *For more information on configuring the driver's window size, refer to* **APDU**.

Utilize ReadPropertyMultiple and WritePropertyMultiple Services
Packing multiple Read/Write operations into a single request can greatly improve performance by reducing the number of transactions required for a given number of tag reads or writes. Check the hardware's PICS document to see if these services are supported. As more items are added to a request, the larger the request and/or response messages become. Large messages may need to be segmented. While it is unlikely that the increased overhead required to send segmented messages would completely negate the performance advantage of using multiple property requests, it is a consideration. Also, when using multiple property requests, the frame and window size issues described above become more relevant.
● *For more information on how to enable multiple property requests, refer to* **APDU**.

COV Reporting
The amount of network traffic and request processing load can be reduced by using Change Of Value (COV) reporting wherever possible.
● *For more information, refer to* **COV Notifications**.

Event Notifications
The amount of network traffic and request processing load can be reduced by using event reporting where applicable. With event notifications enabled, tags that rely on event notification updates (**event-related properties**) refresh the data from cache, which is updated from event notifications. Therefore, the driver does not issue read requests, which allows these tags to be scanned quickly for incoming event notifications. Tags that require polling the device for updates can be set with a scan rate more realistic to the expected rate of change.

🔷 *For more information, refer to **Event Notifications**.*

Use Watchdog Tags in Conjunction with COV or Event Reporting

The OPC quality of tags updated by event and COV notifications may be GOOD, even if the device has momentarily gone offline. Polled tags, by comparison, can quickly reveal a communications problem when an expected read response fails to arrive. When a poll fails, the driver flags the device as being in an error state. The driver uses the device error state to set the quality of event and COV tags. To monitor just event or COV properties, consider polling one additional property in the device to monitor the device communications.

Combining COV Reporting with Event Reporting

When the condition configured to trigger sending an event notification requires the Monitored Value or StatusFlags property of the monitored object to transition from a normal to an off-normal state, the change in value of those properties is not updated while within the normal range. One option, where applicable, is to configure COV on the object as well. In this case, any change in the value is also sent via COV notifications.
🔷 *For more information, refer to **Event Notifications**, **COV**, and **COV Notifications**.*

Configuring Multiple Channels

Multiple channels can increase the driver's performance; however, unlike most Ethernet drivers for the OPC server, the BACnet/IP Driver requires that each channel bind to a unique local address. The address is the combination of IP and port. The local IP used is associated with a Network Adapter Card (NIC) installed on the system, and can be selected with the Network Interface. *For more information on setting the local port, refer to **Network Settings**.*

🔷 *See Also: **Optimizing Communications**.*

Selecting **Default** for network adapter allow the driver to use the operating system's normal bind order to set the NIC that is used. This is the recommended selection for a single channel BACnet/IP Driver project because it may be run on another computer without modification. For clarity of the actual local IP usage, it is not recommended that Default be selected in any of the channels in a multi-channel project.

If a channel is configured with an IP and port combination that is already in use by another BACnet/IP Driver on the channel, a message appears to indicate the problem. This message is not displayed if an invalid adapter selection is written to the "_NetworkAdapter" channel System Tag; however, such configuration changes cause communications with all devices on that channel to fail.

Using Multiple Local IP Addresses

To create a project with multiple channels while using the same UDP port on each, the project will need to be run on a multi-homed computer, which is a computer that has multiple IP addresses associated with it. Each BACnet/IP Driver channel will then bind to a different local IP. A computer can be multi-homed by installing multiple NICs or by associating multiple IP addresses with a single NIC. The process of adding IP addresses to a single NIC system differs slightly depending on the version of Windows being used.

Adding IP Addresses to a Single NIC on Windows

1. Locate and launch the **Control Panel**.

2. Click the **Network** icon and then click on the **Protocols** tab.

3. Select **TCP/IP Protocol**.

4. Click **Properties** and then select the **IP Address** tab.

5. Click **Advanced** then **Add**.

6. Enter the additional IP address and subnet mask.

7. Click **OK**.

8. Restart the computer.

Adding IP Addresses to a Single NIC

1. Click the **My Computer** icon and select **Control Panel**.

2. Click the **Network** and **Dial-Up Connections** icon.

3. Click the **Local Area Connection** icon (or other icon associated with NIC of interest).

4. Click the **Properties** button and select **Internet Protocol (TCP/IP)**.

5. Click **Properties | Advanced**.

6. Select the **IP Settings** tab and click **Add**.

7. Enter the additional IP address and subnet mask.

8. Click **OK**.

Adding IP Addresses to Windows

1. Click **Start** and open **Network Connections**.

2. Next, click **Control Panel | Network and Internet**.

3. Select the connection that is changed.

4. Click **Properties** and provide the administrator password or confirmation (if prompted).

5. Click the **Networking** group. Beneath **This connection uses the following items**, click **Internet Protocol Version 4 (TCP/IPv4)**.

6. Next, click **Properties**. Ensure that the connection is set to use an IP address by clicking **Use the following IP address**.

7. Next, specify the IP address settings in the **IP Address**, **Subnet Mask**, and **Default Gateway** fields.

8. To add a second IP address click **Advanced | IP Settings**. Beneath **IP Address**, click **Add**. Enter a new IP address and subnet mask.

Notes:

- Multi-homing is not supported by all operating system versions.
- Users can only multi-home a network card that is configured to use static IP addresses.
- Some Windows systems support up to five IP addresses for each NIC via the control panel. If more IP addresses are required, add them to the registry manually. To browse, look under HEKY_LOCAL_ MACHINE\SYSTEM\CurrentControlSet\Services. Select the service associated with the adapter card in question. Under the service, go to the Parameters\TCPIP subkey. Then add the IP addresses to IPAddress. Edit SubnetMask and add an entry for each new IP address.
- Although there are additional operating system overhead when running on a multi-homed system, it

will typically be negligible when compared to the performance gained from the use of multiple channels.

Data Types Description

Data Type	Description
Boolean	Single bit
Word	Unsigned 16-bit value bit 0 is the least significant bit bit 15 the most significant bit
Short	Signed 16-bit value bit 0 is the least significant bit bit 14 the most significant bit bit 15 is the sign bit
DWord	Unsigned 32-bit value bit 0 is the least significant bit bit 31 the most significant bit
Long	Signed 32-bit value bit 0 is the least significant bit bit 30 the most significant bit bit 31 is the sign bit
Float	32-bit Floating point value bit 0 is the low bit bit 31 is the high bit
String	Null-terminated ASCII string

Enumerated Data Types

This driver expresses property values with enumerated BACnet data types as integers. The standard enumeration definitions given in the BACnet protocol specification are displayed below.

Enumerated BACnet Data Types

BACnetAction

BACnetBackupState

BACnetBinaryPV

BACnetDeviceStatus

BACnetEngineeringUnits

BACnetEventState

BACnetEventType

BACnetFileAccessMethod

BACnetLifeSafetyMode

BACnetLifeSafetyOperation

BACnetLifeSafetyState

BACnetMaintenance

BACnet NotifyType

BACnet Object Type

BACnet Polarity

BACnet Program Error

BACnet Program Request

BACnet Program State

BACnet Reliability

BACnet Segmentation

BACnet SilencedState

BACnet VTClass

BACnet Action

Value	Action
0	Direct
1	Reverse

BACnetBackupState

Value	Event Type
0	Idle
1	Preparing for backup
2	Preparing for restore
3	Performing a backup
4	Performing a restore
5	Backup failure
6	Restore failure

BACnetBinaryPV

Value	Binary Present Value
0	Inactive
1	Active

BACnetDeviceStatus

Value	Device Status
0	Operational
1	Operational-Read Only
2	Download-required
3	Download-in-progress
4	Non-operational
5	Backup-in-progress

BACnetEngineeringUnits
Acceleration

Value	Unit
166	Meters-per-second-per-second

Area

Value	Unit
0	Square-meters
1	Square-feet
115	Square-inches
116	Square-centimeters

Currency

Value	Unit
105	Currency1
106	Currency2
107	Currency3
108	Currency4
109	Currency5
110	Currency6
111	Currency7
112	Currency8
113	Currency9
114	Currency10

Electrical

Value	Unit
2	Milliamperes
3	Amperes
4	Ohms
5	Volts
6	Kilo-volts
7	Mega-volts
8	Volt-amperes
9	Kilo-volt-amperes
10	Mega-volt-amperes
11	Volt-amperes-reactive
12	Kilo-volt-amperes-reactive
13	Mega-volt-amperes-reactive
14	Degrees-phase
15	Power-factor
122	Kilohms
123	Megohms

Value	Unit
124	Millivolts
145	Milliohms
167	Amperes-per-meter
168	Amperes-per-square-meter
169	Ampere-square-meters
170	Farads
171	Henrys
172	Ohm-meters
173	Siemens
174	Siemens-per-meter
175	Teslas
176	Volts-per-degree-Kelvin
177	Volts-per-meter
178	Webers

Energy

Value	Unit
16	Joules
17	Kilojoules
18	Watt-hours
19	Kilowatt-hours
20	BTUs
21	Therms
22	Ton-hours
125	Kilojoules-per-kilogram
126	Megajoules
146	Megawatt-hours
147	Kilo-BTUs
148	Mega-BTUs

Enthalpy

Value	Unit
23	Joules-per-kilogram-dry-air
24	BTUs-per-pound-dry-air
117	BTUs-per-pound
149	Kilojoules-per-kilogram-dry-air
150	Megajoules-per-kilogram-dry-air

Entrophy

Value	Unit
127	Joules-per-degree-Kelvin
128	Joules-per-kilogram-degree-Kelvin
151	Kilojoules-per-degree-Kelvin
152	Megajoules-per-degree-Kelvin

Force

Value	Unit
153	Newton

Frequency

Value	Unit
25	Cycles-per-hour
26	Cycles-per-minute
27	Hertz
129	Kilohertz
130	Megahertz
131	Per-hour

Humidity

Value	Unit
28	Grams-of-water-per-kilogram-dry-air
29	Percent-relative-humidity

Length

Value	Unit
30	Millimeters
31	Meters
32	Inches
33	Feet
118	Centimeters

Light

Value	Unit
34	Watts-per-square-foot
35	Watts-per-square-meter
36	Lumens
37	Luxes
38	Foot-candles
179	Candelas
180	Candelas-per-square-meter

Mass

Value	Unit
39	Kilograms
40	Pounds-mass
41	Tons

Mass Flow

Value	Unit
42	Kilograms-per-second
43	Kilograms-per-minute
44	Kilograms-per-hour
45	Pounds-mass-per-minute
46	Pounds-mass-per-hour
119	Pounds-mass-per-second
154	Grams-per-second
155	Grams-per-minute
156	Tons-per-hour

Power

Value	Unit
47	Watts
48	Kilowatts
49	Megawatts
50	BTUs-per-hour
51	Horsepower
52	Tons-refrigeration
132	Milliwatts
157	Kilo-BTUs-per-hour

Pressure

Value	Unit
53	Pascals
54	Kilopascals
55	Bars
56	Pounds-force-per-square-inch
57	Centimeters-of-water
58	Inches-of-water
59	Millimeters-of-mercury
60	Centimeters-of-mercury
61	Inches-of-mercury
133	Hectopascals

Value	Unit
134	Millibars

Temperature

Value	Unit
62	Degrees-Celsius
63	Degrees-Kelvin
64	Degrees-Fahrenheit
65	Degree-days-Celsius
66	Degree-days-Fahrenheit
120	Delta-Degrees-Fahrenheit
121	Delta-Degrees-Kelvin
181	Kelvins-per-hour
182	Kelvins-per-minute

Time

Value	Unit
67	Years
68	Months
69	Weeks
70	Days
71	Hours
72	Minutes
73	Seconds
158	Hundredths-seconds
159	Milliseconds

Torque

Value	Unit
160	Newton-meters

Velocity

Value	Unit
74	Meters-per-second
75	Kilometers-per-hour
76	Feet-per-second
77	Feet-per-minute
78	Miles-per-hour
161	Millimeters-per-second
162	Millimeters-per-minute
163	Meters-per-minute

Value	Unit
164	Meters-per-hour

Volume

Value	Unit
79	Cubic-feet
80	Cubic-meters
81	Imperial-gallons
82	Liters
83	Us-gallons

Volumetric Flow

Value	Unit
84	Cubic-feet-per-minute
85	Cubic-meters-per-second
86	Imperial-gallons-per-minute
87	Liters-per-second
88	Liters-per-minute
89	Us-gallons-per-minute
135	Cubic-meters-per-hour
136	Liters-per-hour
142	Cubic-feet-per-second
165	Cubic-meters-per-minute

Other

Value	Unit
90	Degrees-angular
91	Degrees-Celsius-per-hour
92	Degrees-Celsius-per-minute
93	Degrees-Fahrenheit-per-hour
94	Degrees-Fahrenheit-per-minute
95	No-units
96	Parts-per-million
97	Parts-per-billion
98	Percent
99	Percent-per-second
100	Per-minute
101	Per-second
102	Psi-per-Degree-Fahrenheit
103	Radians
104	Revolutions-per-minute

Value	Unit
137	Kilowatt-hours-per-square-meter
138	Kilowatt-hours-per-square-foot
139	Megajoules-per-square-meter
140	Megajoules-per-square-foot
141	Watts-per-square-meter-Degree-Kelvin
143	Percent-obscuration-per-foot
144	Percent-obscuration-per-meter
183	Joule-seconds
185	Square-meters-per-Newton
186	Kilogram-per-cubic-meter
187	Newton-seconds
188	Newtons-per-meter
189	Watts-per-meter-per-degree-Kelvin

BACnetEventState

Value	Event State
0	Normal
1	Fault
2	Off-normal
3	High-limit
4	Low-limit
5	Life-safety-alarm

BACnetEventType

Value	Event Type
0	Change-of-bitstring
1	Change-of-state
2	Change-of-value
3	Command-failure
4	Floating-limit
5	Out-of-range
6	Complex-event-type
7	Deprecated
8	Change-of-life-safety
9	Extended
10	Buffer-ready
11	Unsigned-range
12	Reserved for future addenda
13	Access-event

Value	Event Type
14	Double-out-of-range
15	Signed-out-of-range
16	Unsigned-out-of-range
17	Change-of-characterstring
18	Change-of-status-flags
19	Change-of-reliability
20	None

BACnetFileAccessMethod

Value	Access Method
0	Record-access
1	Stream-access

BACnetLifeSafetyMode

Value	Life Safety Mode
0	Off
1	On
2	Test
3	Manned
4	Unmanned
5	Armed
6	Disarmed
7	Pre-armed
8	Slow
9	Fast
10	Disconnected
11	Enabled
12	Disabled
13	Automatic-release-disabled
14	Default

BACnetLifeSafetyOperation

Value	Life Safety Operation
0	None
1	Silence
2	Silence-audible
3	Silence-visible
4	Reset
5	Reset-alarm

Value	Life Safety Operation
6	Reset-fault
7	Unsilence
8	Unsilence-audible
9	Unsilence-visual

BACnetLifeSafetyState

Value	Life Safety State
0	Quiet
1	Pre-alarm
2	Alarm
3	Fault
4	Fault-pre-alarm
5	Fault-alarm
6	Not-ready
7	Active
8	Tamper
9	Test-alarm
10	Test-active
11	Test-fault
12	Test-fault-alarm
13	Holdup
14	Duress
15	Tamper-alarm
16	Abnormal
17	Emergency-power
18	Delayed
19	Blocked
20	Local-alarm
21	General-alarm
22	Supervisory
23	Test-supervisory

BACnetMaintenance

Value	Maintenance
0	None
1	Periodic-test
2	Need-service-operational
3	Need-service-inoperative

BACnetNotifyType

Value	Notify Type
0	Alarm
1	Event
2	Ack-notification

BACnet ObjectType

Value	Object Type
0	Analog-input
1	Analog-output
2	Analog-value
3	Binary-input
4	Binary-output
5	Binary-value
6	Calendar
7	Command
8	Device
9	Event-enrollment
10	File
11	Group
12	Loop
13	Multi-state-input
14	Multi-state-output
15	Notification-class
16	Program
17	Schedule
18	Averaging
19	Multi-state-value
20	Trend-log
21	Life-safety-point
22	Life-safety-zone
23	Accumulator

BACnet Polarity

Value	Polarity
0	Normal
1	Reverse

BACnet Program Error

Value	Program Error
0	Normal

Value	Program Error
1	Load-failed
2	Internal
3	Program
4	Other

BACnetProgramRequest

Value	Program Request
0	Ready
1	Load
2	Run
3	Halt
4	Restart
5	Unload

BACnetProgramState

Value	Program State
0	Idle
1	Loading
2	Running
3	Waiting
4	Halted
5	Unloading

BACnetReliability

Value	Reliability
0	No-fault-detected
1	No-sensor
2	Over-range
3	Under-range
4	Open-loop
5	Shorted-loop
6	No-output
7	Unreliable-other
8	Process-error
9	Multi-state-fault
10	Configuration-error
12	Communication-failure
13	Member-fault
14	Monitored-object-fault
15	Tripped

BACnet Segmentation

Value	Segmentation
0	Segmented-both
1	Segmented-transmit
2	Segmented-receive
3	No-segmentation

BACnet SilencedState

Value	Silenced State
0	Unsilenced
1	Audible-silenced
2	Visible-silenced
3	All-silenced

BACnet VTClass

Value	VT Class
0	Default-terminal
1	ANSI-x3-64
2	Dec-vt52
3	Dec-vt100
4	Dec-vt220
5	Hp-700-94
6	IBM-3130

Address Descriptions

All addresses have three required fields: object type, object instance, and property identifier. Additional fields may be required for some properties. Many addresses may take optional fields.

Basic Addresses (Primitive Data Types)

Properties with primitive data types are addressed using the following format:

<object type>.<object instance>.<property identifier>

- The **object type** field contains a mnemonic from the list of supported BACnet objects.
- The **object instance** field is the numerical object instance. Object instances may range from 0 to 4194303.
- The **property identifier** field contains the mnemonic for a property that is a member of the selected object type.

💠 *See Also*: *BACnet/IP Objects*

BACnet Array and List Addresses

Elements of arrays and lists are addressed using the following format:

<object type>.<object instance>.<property identifier>[index]

This data exists in array or list form in the BACnet/IP device, and not the OPC server. One tag must be configured for each array or list element. The data is not presented to the OPC clients in array form because BACnet array and list elements may not have primitive data types. The driver attempts to optimize reads of array data, meaning that it generally reads data for all referenced elements in a single transaction. BACnet lists must be read in their entirety, regardless of the number of elements needed. Element indices start at 1. The upper limit depends on both the property for arrays and the device configuration for lists.

Complex Addresses (Structured Data Types)

Elements of structured data types are addressed using the following format:

<object type>.<object instance>.<property identifier>.<sub property 1>.<sub property 2> ...

- The **sub property n** fields are one of the mnemonics given in the links from the supported object types. *For more information, refer to **BACnet/IP Objects**.*

Address Modifiers

Optional address modifiers may be added to alter the behavior of the driver. The property address and modifier must be separated by a single space character. The available address modifiers are as follows:

- **COV**: If this modifier is present, the driver attempts to subscribe to change of value (COV) notifications for the addressed property. Use of COV notifications over polling can greatly reduce network traffic. The BACnet specification requires that certain properties must support COV, but does not demand that others cannot. A particular device may offer COV support for any property. This modifier is primarily for these non-standard COV properties. The driver can be configured to assume COV is to be used for certain standard COV properties, regardless of the presence of this modifier. This behavior can be turned on or off with the COV mode device setting. *For more information, refer to the device's PICS statement.*

- **RELINQUISH**: BACnet requires that a device execute writes according to a command priority. Once a write has been executed, the issuing application retains "command" over that property. That is, no other application can write to that property unless it uses a higher priority, or the "commanding" application "relinquishes" its command over the property. A tag with this address modifier can be used to relinquish command over the addressed property. Such tags are Write Only, and have a default data type of Boolean. The driver issues a relinquish command request when any value is written to this tag. Command over other properties is not affected.

- **PRIORITY**: This modifier, when appended to a tag address for the present value property of a standard commandable object, provides object-level priority for this commandable object. A write to this tag, valid in the range of 0 – 16 inclusively, sets the object's command priority. A value of 0 indicates this object is defaulting to the device-level priority. Other valid values specify the object-level priority. Tags with this address and modifier are read/write and have a default data type of Short. Writes to this tag are managed by the driver and persist in memory until the runtime shuts down. Tags with this address default to 0.

🔷 **See Also**: *BACnet/IP Objects*

Addressing Examples

The following examples assume an analog value object with instance number 100. *For more information on the object, refer to **Analog Value**.*

1. **AnalogValue.100.PresentValue** addresses the **Present Value** property. Since this property has "implicit COV," the driver may subscribe to COV notifications of this property or continuously poll for its current value.

2. **AnalogValue.100.OutOfService COV** addresses the **Out Of Service** property. The COV address modifier is used to tell the driver that COV reporting can be used for this property, even though this property does not normally support COV.

3. **AnalogValue.100.PresentValue RELINQUISH** address is used to create a Write Only tag for relinquishing the driver's command over the **PresentValue** property. *For more information, refer to **[Command Settings](#)**.*

4. **AnalogValue.100.PresentValue PRIORITY** address is used to create a Read/Write tag for setting object-level priority for a standard commandable object. *For more information, refer to **[Command Settings](#)**.*

5. **AnalogValue.100.PriorityArray [4]** addresses element 4 of the **Priority Array**.

6. **AnalogValue.100.EventEnable.ToFault** addresses the **To Fault** element of the **Event Enable** bit string property.

7. **AnalogValue.100.EventEnable** addresses all bits of the Event Enable bit string property, packed as Word value. Only the lowest 3 bits of the Word is meaningful in this case. These is **ToOffNormal**, **ToFault**, and **ToNormal** respectively.

8. **AnalogValue.100.ObjectIdentifier.ObjectInstance** addresses the **Object Instance** member of the **Object Identifier** property structure.

9. **AnalogValue.100.ObjectIdentifier** addresses the Object Identifier property structure, and packs its member values into a single DWord value. The high 10 bits are the **Object Type** member, and the low 22 bits are the Object Instance member.

◆ *For more information on enabling COV in the BACnet/IP Driver, refer to **[COV Notifications](#)** and **[COV Settings](#)**.*

BACnet/IP Objects

For more information on a specific BACnet/IP object, select a link from the list below.

[Accumulator](#)
[Analog Input](#)
[Analog Output](#)
[Analog Value](#)
[Averaging](#)
[Binary Input](#)
[Binary Output](#)
[Binary Value](#)
[Calendar](#)
[Command](#)
[Device](#)
[Event Enrollment](#)
[File](#)
[Group](#)

Life Safety Point
Life Safety Zone
Loop
Multi-State Input
Multi-State Output
Multi-State Value
Notification Class
Program
Schedule
Trend Log

Accumulator

The following table describes the object's supported properties. The Access column specifies the default access permission for tags. To accommodate non-standard implementations of BACnet, tags may be given Read/Write access unless noted otherwise. The COV column specifies whether the driver considers the property to have implicit Change Of Value (COV) notification capability; that is, whether the BACnet specification requires the property to support COV. For some properties, COV support depends on implementation. The "COV" modifier must be added to the tag's address for use. *For more information, refer to* **COV Settings**.

🔘 **Note**: The length of the array property is specified by *[m]*, where m is the number of supported elements (according to the BACnet specification). The BACnet array properties that do not have a length specified by the BACnet standard is designated by *[N]*. This means the length of the property array depends on the BACnet device. *For more information, refer to* **Addressing Examples**.

🔷 **See Also**: **Address Descriptions**

Property Mnemonic	BACnet Data Type	OPC Data Type	Access	COV
AckedTransitions[1] .ToFault .ToNormal .ToOffNormal	BACnetEventTransitionBits	**Word**, Short Boolean Boolean Boolean	Read Only	No
Description	CharacterString	**String**	Read Only	No
DeviceType	CharacterString	**String**	Read Only	No
EventEnable[1] .ToFault .ToNormal .ToOffNormal	BACnetEventTransitionBits	**Word**, Short Boolean Boolean Boolean	Read/Write	No
EventState	**BACnetEventState**[2]	**DWord**, Long	Read Only	No
EventTimeStamps[3]	Array of Timestamp	**String**	Read Only[3]	No
HighLimit	Unsigned Integer	**DWord**, Long	Read Only	No
LimitEnable[1] .HighLimitEnable .LowLimitEnable	BACnetLimitEnable	**Word**, Short Boolean Boolean	Read/Write	No

Property Mnemonic	BACnet Data Type	OPC Data Type	Access	COV
LimitMonitoringInterval	Unsigned Integer	**DWord**, Long	Read/Write	No
LowLimit	Unsigned Integer	**DWord**, Long	Read Only	No
MaxPresValue	Unsigned Integer	**DWord**, Long	Read Only	No
NotificationClass	Unsigned Integer	**DWord**, Long	Read Only	No
NotifyType	**BACnetNotifyType**[2]	**DWord**, Long	Read Only	No
ObjectIdentifier[4] .ObjectInstance .ObjectType	BACnetObjectIdentifier	**DWord**, Long **DWord**, Long **DWord**, Long	Read Only[3]	No
ObjectName	CharacterString	**String**	Read Only	No
ObjectType	**BACnetObjectType**[2]	**DWord**, Long	Read Only[3]	No
OutOfService	Boolean	**Boolean**	Read/Write	No
Prescale[5]	BACnetPrescale	**String**	Read/Write	No
PresentValue	Unsigned Integer	**DWord**, Long	Read/Write	No
ProfileName	CharacterString	**String**	Read Only	No
PulseRate	Unsigned Integer	**DWord**, Long	Read Only	No
Reliability	**BACnetReliability**[2]	**DWord**, Long	Read Only	No
Scale[6]	BACnetScale	**String**	Read/Write	No
StatusFlags[1] .Fault .InAlarm .OutOfService .Overridden	BACnetStatusFlags	**Word**, Short Boolean Boolean Boolean Boolean	Read Only	No
TimeDelay	Unsigned Integer	**DWord**, Long	Read/Write	No
Units	**BACnetEngineeringUnits**[2]	**DWord**, Long	Read Only	No
ValueBeforeChange	Unsigned Integer	**DWord**, Long	Read Only	No
ValueChangeTime	BACnetDateTime	**String**	Read Only[3]	No
ValueSet	Unsigned Integer	**DWord**, Long	Read Only	No

Notes:

1. Bit string types may be viewed as a packed Word value, whose actual number of meaningful bits depends on the specific property. They may also be viewed as individual bits using optional sub-property fields.

2. Properties with enumerated BACnet Data Types are expressed as integer values. For standard interpretations, refer to **Enumerated Data Types**.

3. May not be made writable.

4. Object Identifier may be viewed as a packed DWord value (high 10 bits are the object type and low 22 bits are the object instance) or as individual tags for object type and instance using optional sub-property fields.

See Also:

1. *For more information, refer to the* **Prescale String Format***.*

2. *For more information, refer to the* **Scale String Format***.*

Priority Array Elements

Priority Array elements may be "NULL" or the numerical command value currently in effect. The array element index may range from 1 to 16, inclusive.

Analog Input

The following table describes the object's supported properties. The Access column specifies the default access permission for tags. To accommodate non-standard implementations of BACnet, tags may be given Read/Write access unless noted otherwise. The COV column specifies whether the driver considers the property to have implicit Change Of Value (COV) notification capability; that is, whether the BACnet specification requires the property to support COV. For some properties, COV support depends on implementation. The "COV" modifier must be added to the tag's address for use. *For more information, refer to COV Settings*.

 Note: The length of the array property is specified by *[m]*, where m is the number of supported elements (according to the BACnet specification). The BACnet array properties that do not have a length specified by the BACnet standard is designated by *[N]*. This means the length of the property array depends on the BACnet device.

 For more information, refer to Address Descriptions, Addressing Examples.

Property Mnemonic	BACnet Data Type	OPC Data Type	Access	COV
AckedTransitions[1] .ToFault .ToNormal .ToOffNormal	BACnetEventTransitionBits	**Word**, Short Boolean Boolean Boolean	Read Only	No
COVIncrement	REAL	**Float**	Read/Write	No
Deadband	REAL	**Float**	Read/Write	No
Description	CharacterString	**String**	Read Only	No
DeviceType	CharacterString	**String**	Read Only	No
EventEnable[1] .ToFault .ToNormal .ToOffNormal	BACnetEventTransitionBits	**Word**, Short Boolean Boolean Boolean	Read/Write	No
EventState	**BACnetEventState**[2]	**DWord**, Long	Read Only	No
EventTimeStamps[3]	Array of Timestamp	**String**	Read Only[3]	No
HighLimit	REAL	**Float**	Read Only	No
LimitEnable[1] .HighLimitEnable .LowLimitEnable	BACnetLimitEnable	**Word**, Short Boolean Boolean	Read/Write	No
LowLimit	REAL	**Float**	Read Only	No
MaxPresValue	REAL	**Float**	Read Only	No
MinPresValue	REAL	**Float**	Read Only	No
NotificationClass	Unsigned Integer	**DWord**, Long	Read Only	No
NotifyType	**BACnetNotifyType**[2]	**DWord**, Long	Read Only	No
ObjectIdentifier[4] .ObjectInstance .ObjectType	BACnetObjectIdentifier	**DWord**, Long **DWord**, Long **DWord**, Long	Read Only[3]	No

Property Mnemonic	BACnet Data Type	OPC Data Type	Access	COV
ObjectName	CharacterString	**String**	Read Only	No
ObjectType	**BACnetObjectType**[2]	**DWord**, Long	Read Only[3]	No
OutOfService	Boolean	**Boolean**	Read/Write	No
PresentValue	REAL	**Float**	Read/Write	Yes
ProfileName	CharacterString	**String**	Read Only	No
Reliability	**BACnetReliability**[2]	**DWord**, Long	Read Only	No
Resolution	REAL	**Float**	Read Only	No
StatusFlags[1] .Fault .InAlarm .OutOfService .Overridden	BACnetStatusFlags	**Word**, Short Boolean Boolean Boolean Boolean	Read Only	Yes
TimeDelay	Unsigned Integer	**DWord**, Long	Read/Write	No
Units	**BACnetEngineeringUnits**[2]	**DWord**, Long	Read Only	No
UpdateInterval	Unsigned Integer	**DWord**, Long	Read/Write	No

Notes:

1. Bit string types may be viewed as a packed Word value, whose actual number of meaningful bits depends on the specific property. They may also be viewed as individual bits using optional sub-property fields.

2. Properties with enumerated BACnet Data Types are expressed as integer values. For standard interpretations, refer to **Enumerated Data Types**.

3. May not be made writable.

4. Object Identifier may be viewed as a packed DWord value (high 10 bits are the object type and low 22 bits are the object instance) or as individual tags for object type and instance using optional sub-property fields.

Priority Array Elements

Priority Array elements may be "NULL" or the numerical command value currently in effect. The array element index may range from 1 to 16, inclusive.

Analog Output

The following table describes the object's supported properties. The Access column specifies the default access permission for tags. To accommodate non-standard implementations of BACnet, tags may be given Read/Write access unless noted otherwise. The COV column specifies whether the driver considers the property to have implicit Change Of Value (COV) notification capability; that is, whether the BACnet specification requires the property to support COV. For some properties, COV support depends on implementation. The "COV" modifier must be added to the tag's address for use. *For more information, refer to COV Settings*.

Note: The length of the array property is specified by *[m]*, where m is the number of supported elements (according to the BACnet specification). The BACnet array properties that do not have a length specified by

the BACnet standard is designated by *[N]*. This means the length of the property array depends on the BACnet device. *For more information, refer to **Addressing Examples**.*

🔷 *For more information, refer to **Address Descriptions, Addressing Examples.***

Property Mnemonic	BACnet Data Type	OPC Data Type	Access	COV
AckedTransitions[1] .ToFault .ToNormal .ToOffNormal	BACnetEventTransitionBits	**Word**, Short Boolean Boolean Boolean	Read Only	No
COVIncrement	REAL	**Float**	Read/Write	No
Deadband	REAL	**Float**	Read/Write	No
Description	CharacterString	**String**	Read Only	No
DeviceType	CharacterString	**String**	Read Only	No
EventEnable[1] .ToFault .ToNormal .ToOffNormal	BACnetEventTransitionBits	**Word**, Short Boolean Boolean Boolean	Read/Write	No
EventState	**BACnetEventState**[2]	**DWord**, Long	Read Only	No
EventTimestamps[3]	Array of Timestamp	**String**	Read Only[3]	No
HighLimit	REAL	**Float**	Read Only	No
LimitEnable[1] .HighLimitEnable .LowLimitEnable	BACnetLimitEnable	**Word**, Short Boolean Boolean	Read/Write	No
LowLimit	REAL	**Float**	Read Only	No
MaxPresValue	REAL	**Float**	Read Only	No
MinPresValue	REAL	**Float**	Read Only	No
NotificationClass	Unsigned Integer	**DWord**, Long	Read Only	No
NotifyType	**BACnetNotifyType**[2]	**DWord**, Long	Read Only	No
ObjectIdentifier[4] .ObjectInstance .ObjectType	BACnetObjectIdentifier	**DWord**, Long **DWord**, Long **DWord**, Long	Read Only[3]	No
ObjectName	CharacterString	**String**	Read Only	No
ObjectType	**BACnetObjectType**[2]	**DWord**, Long	Read Only[3]	No
OutOfService	Boolean	**Boolean**	Read/Write	No
PresentValue	REAL	**Float**	Read/Write	Yes
PriorityArray[16]	Array of BACnetPriorityArray	**String**	Read Only[3]	No
ProfileName	CharacterString	**String**	Read Only	No
Reliability	**BACnetReliability**[2]	**DWord**, Long	Read Only	No
RelinquishDefault	REAL	**Float**	Read/Write	No

Property Mnemonic	BACnet Data Type	OPC Data Type	Access	COV
Resolution	REAL	**Float**	Read Only	No
StatusFlags[1] .Fault .InAlarm .OutOfService .Overridden	BACnetStatusFlags	**Word**, Short Boolean Boolean Boolean Boolean	Read Only	Yes
TimeDelay	Unsigned Integer	**DWord**, Long	Read/Write	No
Units	**BACnetEngineeringUnits**[2]	**DWord**, Long	Read Only	No

⬤ **Notes**:

1. Bit string types may be viewed as a packed Word value, whose actual number of meaningful bits depends on the specific property. They may also be viewed as individual bits using optional sub-property fields.

2. Properties with enumerated BACnet Data Types are expressed as integer values. For standard interpretations, refer to **Enumerated Data Types**.

3. May not be made writable.

4. Object Identifier may be viewed as a packed DWord value (high 10 bits are the object type and low 22 bits are the object instance) or as individual tags for object type and instance using optional sub-property fields.

Priority Array Elements

Priority Array elements may be "NULL" or the numerical command value currently in effect. The array element index may range from 1 to 16, inclusive.

Analog Value

The following table describes the object's supported properties. The Access column specifies the default access permission for tags. To accommodate non-standard implementations of BACnet, tags may be given Read/Write access unless noted otherwise. The COV column specifies whether the driver considers the property to have implicit Change Of Value (COV) notification capability; that is, whether the BACnet specification requires the property to support COV. For some properties, COV support depends on implementation. The "COV" modifier must be added to the tag's address for use. *For more information, refer to **COV Settings**.*

⬤ **Note**: The length of the array property is specified by *[m]*, where m is the number of supported elements (according to the BACnet specification). The BACnet array properties that do not have a length specified by the BACnet standard is designated by *[N]*. This means the length of the property array depends on the BACnet device.

⬤ *For more information, refer to **Address Descriptions**, **Addressing Examples**.*

Property Mnemonic	BACnet Data Type	OPC Data Type	Access	COV
AckedTransitions[1] .ToFault	BACnetEventTransitionBits	**Word**, Short Boolean	Read Only	No

Property Mnemonic	BACnet Data Type	OPC Data Type	Access	COV
.ToNormal .ToOffNormal		Boolean Boolean		
COVIncrement	REAL	**Float**	Read/Write	No
Deadband	REAL	**Float**	Read/Write	No
Description	CharacterString	**String**	Read Only	No
EventEnable[1] .ToFault .ToNormal .ToOffNormal	BACnetEventTransitionBits	**Word**, Short Boolean Boolean Boolean	Read/Write	No
EventState	**BACnetEventState**[2]	**DWord**, Long	Read Only	No
EventTimeStamps[3]	Array of TimeStamp	**String**	Read Only[3]	No
HighLimit	REAL	**Float**	Read Only	No
LimitEnable[1] .HighLimitEnable .LowLimitEnable	BACnetLimitEnable	**Word**, Short Boolean Boolean	Read/Write	No
LowLimit	REAL	**Float**	Read Only	No
NotificationClass	Unsigned Integer	**DWord**, Long	Read Only	No
NotifyType	**BACnetNotifyType**[2]	**DWord**, Long	Read Only	No
ObjectIdentifier[4] .ObjectInstance .ObjectType	BACnetObjectIdentifier	**DWord**, Long **DWord**, Long **DWord**, Long	Read Only[3]	No
ObjectName	CharacterString	**String**	Read Only	No
ObjectType	**BACnetObjectType**[2]	**DWord**, Long	Read Only[3]	No
OutOfService	Boolean	**Boolean**	Read/Write	No
PresentValue	REAL	**Float**	Read/Write	Yes
PriorityArray[16]	Array of BACnetPriorityArray	**String**	Read Only[3]	No
ProfileName	CharacterString	**String**	Read Only	No
Reliability	**BACnetReliability**[2]	**DWord**, Long	Read Only	No
RelinquishDefault	REAL	**Float**	Read/Write	No
StatusFlags[1] .Fault .InAlarm .OutOfService .Overridden	BACnetStatusFlags	**Word**, Short Boolean Boolean Boolean Boolean	Read Only	Yes
TimeDelay	Unsigned Integer	**DWord**, Long	Read/Write	No
Units	**BACnetEngineeringUnits**[2]	**DWord**, Long	Read Only	No

Notes:

1. Bit string types may be viewed as a packed Word value, whose actual number of meaningful bits depends on the specific property. They may also be viewed as individual bits using optional sub-property fields.

2. Properties with enumerated BACnet Data Types are expressed as integer values. For standard interpretations, refer to **Enumerated Data Types**.

3. May not be made writable.

4. Object Identifier may be viewed as a packed DWord value (high 10 bits are the object type and low 22 bits are the object instance) or as individual tags for object type and instance using optional sub-property fields.

Priority Array Elements

Priority Array elements may be "NULL" or the numerical command value currently in effect. The array element index may range from 1 to 16, inclusive.

Averaging

The following table describes the object's supported properties. The Access column specifies the default access permission for tags. To accommodate non-standard implementations of BACnet, tags may be given Read/Write access unless noted otherwise.

Notes:

1. The length of the array property is specified by [m], where m is the number of supported elements (according to the BACnet specification). The BACnet array properties that do not have a length specified by the BACnet standard is designated by [N]. This means the length of the property array depends on the BACnet device. *For more information, refer to* **Addressing Examples**.

2. Implicit Change of Value (COV) notifications are not supported for this object.

For more information, refer to **Address Descriptions**, **Addressing Examples**, **COV Notifications**.

Property Mnemonic	BACnet Data Type	OPC Data Type	Access
AttemptedSamples	Unsigned Integer	**DWord**, Long	Read Only
AverageValue	REAL	**Float**	Read Only
Description	CharacterString	**String**	Read Only
MaximumValue	REAL	**Float**	Read Only
MaximumValueTimestamp	BACnetDateTime	**String**	Read Only[1]
MinimumValue	REAL	**Float**	Read Only
MinimumValueTimestamp	BACnetDateTime	**String**	Read Only[1]
ObjectIdentifier[2] .ObjectInstance .ObjectType	BACnetObjectIdentifier	**DWord**, Long **DWord**, Long **DWord**, Long	Read Only[1]
ObjectName	CharacterString	**String**	Read Only
ObjectType	**BACnetObjectType**[3]	**DWord**, Long	Read Only[1]

Property Mnemonic	BACnet Data Type	OPC Data Type	Access
ProfileName	CharacterString	**String**	Read Only
ValidSamples	Unsigned Integer	**DWord**, Long	Read Only
VarianceValue	REAL	**Float**	Read Only
WindowInterval	Unsigned Integer	**DWord**, Long	Read/Write
WindowSamples	Unsigned Integer	**DWord**, Long	Read/Write

Notes:

1. May not be made writable.

2. Object Identifier may be viewed as a packed DWord value (high 10 bits are the object type and low 22 bits are the object instance) or as individual tags for object type and instance using optional sub-property fields.

3. Properties with enumerated BACnet Data Types are expressed as integer values. For standard interpretations, refer to **Enumerated Data Types**.

Priority Array Elements

Priority Array elements may be "NULL" or the numerical command value currently in effect. The array element index may range from 1 to 16, inclusive.

Binary Input

The following table describes the object's supported properties. The Access column specifies the default access permission for tags. To accommodate non-standard implementations of BACnet, tags may be given Read/Write access unless noted otherwise. The COV column specifies whether the driver considers the property to have implicit Change Of Value (COV) notification capability; that is, whether the BACnet specification requires the property to support COV. For some properties, COV support depends on implementation. The "COV" modifier must be added to the tag's address for use. *For more information, refer to **COV Settings**.*

Note: The length of the array property is specified by *[m]*, where m is the number of supported elements (according to the BACnet specification). The BACnet array properties that do not have a length specified by the BACnet standard is designated by *[N]*. This means the length of the property array depends on the BACnet device.

*For more information, refer to **Address Descriptions, Addressing Examples.***

Property Mnemonic	BACnet Data Type	OPC Data Type	Access	COV
AckedTransitions[1] .ToFault .ToNormal .ToOffNormal	BACnetEventTransitionBits	**Word**, Short Boolean Boolean Boolean	Read Only	No
ActiveText	CharacterString	**String**	Read Only	No
AlarmValue	**BACnetBinaryPV**[2]	**Boolean**	Read/Write	No
ChangeOfStateCount	Unsigned Integer	**DWord**, Long	Read/Write	No
ChangeOfStateTime	BACnetDataTime	**String**	Read Only[3]	No

Property Mnemonic	BACnet Data Type	OPC Data Type	Access	COV
Description	CharacterString	**String**	Read Only	No
DeviceType	CharacterString	**String**	Read Only	No
ElapsedActiveTime	Unsigned Integer	**DWord**, Long	Read Only	No
EventEnable[1] .ToFault .ToNormal .ToOffNormal	BACnetEventTransitionBits	**Word**, Short Boolean Boolean Boolean	Read/Write	No
EventState	**BACnetEventState**[2]	**DWord**, Long	Read Only	No
EventTimeStamps[3]	Array of TimeStamp	**String**	Read Only[3]	No
InactiveText	CharacterString	**String**	Read Only	No
NotificationClass	Unsigned Integer	**DWord**, Long	Read Only	No
NotifyType	**BACnetNotifyType**[2]	**DWord**, Long	Read Only	No
ObjectIdentifier[4] .ObjectInstance .ObjectType	BACnetObjectIdentifier	**DWord**, Long **DWord**, Long **DWord**, Long	Read Only[3]	No
ObjectName	CharacterString	**String**	Read Only	No
ObjectType	**BACnetObjectType**[2]	**DWord**, Long	Read Only[3]	No
OutOfService	Boolean	**Boolean**	Read/Write	No
Polarity	**BACnetPolarity**[2]	**Boolean**	Read/Write	No
PresentValue	**BACnetBinaryPV**[2]	**Boolean**	Read/Write	Yes
ProfileName	CharacterString	**String**	Read Only	No
Reliability	**BACnetReliability**[2]	**DWord**, Long	Read Only	No
StatusFlags[1] .Fault .InAlarm .OutOfService .Overridden	BACnetStatusFlags	**Word**, Short Boolean Boolean Boolean Boolean	Read Only	Yes
TimeDelay	Unsigned Integer	**DWord**, Long	Read/Write	No
TimeOfActiveTimeReset	BACnetDateTime	**String**	Read Only[3]	No
TimeOfStateCountReset	BACnetDateTime	**String**	Read Only[3]	No

Notes:

1. Bit string types may be viewed as a packed Word value, whose actual number of meaningful bits depends on the specific property. They may also be viewed as individual bits using optional sub-property fields.

2. Properties with enumerated BACnet Data Types are expressed as integer values. For standard interpretations, refer to **Enumerated Data Types**.

3. May not be made writable.

4. Object Identifier may be viewed as a packed DWord value (high 10 bits are the object type and low 22 bits are the object instance) or as individual tags for object type and instance using optional sub-property fields.

Priority Array Elements

Priority Array elements may be "NULL" or the numerical command value currently in effect. The array element index may range from 1 to 16, inclusive.

Binary Output

The following table describes the object's supported properties. The Access column specifies the default access permission for tags. To accommodate non-standard implementations of BACnet, tags may be given Read/Write access unless noted otherwise. The COV column specifies whether the driver considers the property to have implicit Change Of Value (COV) notification capability; that is, whether the BACnet specification requires the property to support COV. For some properties, COV support depends on implementation. The "COV" modifier must be added to the tag's address for use. *For more information, refer to* **COV Settings**.

Note: The length of the array property is specified by *[m]*, where m is the number of supported elements (according to the BACnet specification). The BACnet array properties that do not have a length specified by the BACnet standard is designated by *[N]*. This means the length of the property array depends on the BACnet device.

For more information, refer to **Address Descriptions**, **Addressing Examples**.

Property Mnemonic	BACnet Data Type	OPC Data Type	Access	COV
AckedTransitions[1] .ToFault .ToNormal .ToOffNormal	BACnetEventTransitionBits	**Word**, Short Boolean Boolean Boolean	Read Only	No
ActiveText	CharacterString	**String**	Read Only	No
ChangeOfStateCount	Unsigned Integer	**DWord**, Long	Read/Write	No
ChangeOfStateTime	BACnetDateTime	**String**	Read Only[2]	No
Description	CharacterString	**String**	Read Only	No
DeviceType	CharacterString	**String**	Read Only	No
ElapsedActiveTime	Unsigned Integer	**DWord**, Long	Read Only	No
EventEnable[1] .ToFault .ToNormal .ToOffNormal	BACnetEventTransitionBits	**Word**, Short Boolean Boolean Boolean	Read/Write	No
EventState	**BACnetEventState**[3]	**DWord**, Long	Read Only	No
EventTimeStamps[3]	Array of TimeStamp	**String**	Read Only[2]	No
FeedbackValue	**BACnetBinaryPV**[3]	**Boolean**	Read/Write	No
InactiveText	CharacterString	**String**	Read Only	No
MinimumOffTime	Unsigned Integer	**DWord**, Long	Read Only	No

Property Mnemonic	BACnet Data Type	OPC Data Type	Access	COV
MinimumOnTime	Unsigned Integer	**DWord**, Long	Read Only	No
NotificationClass	Unsigned Integer	**DWord**, Long	Read Only	No
NotifyType	**BACnetNotifyType**[3]	**DWord**, Long	Read Only	No
ObjectIdentifier[4]	BACnetObjectIdentifier	**DWord**, Long	Read Only[2]	No
.ObjectInstance		**DWord**, Long		
.ObjectType		**DWord**, Long		
ObjectName	CharacterString	**String**	Read Only	No
ObjectType	**BACnetObjectType**[3]	**DWord**, Long	Read Only[2]	No
OutOfService	Boolean	**Boolean**	Read/Write	No
Polarity	**BACnetPolarity**[3]	**Boolean**	Read/Write	No
PresentValue	**BACnetBinaryPV**[3]	**Boolean**	Read/Write	Yes
PriorityArray[16]	Array of BACnetPriorityArray	**String**	Read Only[2]	No
ProfileName	CharacterString	**String**	Read Only	No
Reliability	**BACnetReliability**[3]	**DWord**, Long	Read Only	No
RelinquishDefault	**BACnetBinaryPV**[3]	**Boolean**	Read/Write	No
StatusFlags[1]	BACnetStatusFlags	**Word**, Short	Read Only	Yes
.Fault		Boolean		
.InAlarm		Boolean		
.OutOfService		Boolean		
.Overridden		Boolean		
TimeDelay	Unsigned Integer	**DWord**, Long	Read/Write	No
TimeOfActiveTimeReset	BACnetDateTime	**String**	Read Only[2]	No
TimeOfStateCountReset	BACnetDateTime	**String**	Read Only[2]	No

Notes:

1. Bit string types may be viewed as a packed Word value, whose actual number of meaningful bits depends on the specific property. They may also be viewed as individual bits using optional sub-property fields.

2. May not be made writable.

3. Properties with enumerated BACnet Data Types are expressed as integer values. For standard interpretations, refer to **Enumerated Data Types**.

4. Object Identifier may be viewed as a packed DWord value (high 10 bits are the object type and low 22 bits are the object instance) or as individual tags for object type and instance using optional sub-property fields.

Priority Array Elements

Priority Array elements may be "NULL" or the numerical command value currently in effect. The array element index may range from 1 to 16, inclusive.

Binary Value

The following table describes the object's supported properties. The Access column specifies the default access permission for tags. To accommodate non-standard implementations of BACnet, tags may be given Read/Write access unless noted otherwise. The COV column specifies whether the driver considers the property to have implicit Change Of Value (COV) notification capability; that is, whether the BACnet specification requires the property to support COV. For some properties, COV support depends on implementation. The "COV" modifier must be added to the tag's address for use. *For more information, refer to **COV Settings**.*

⬤ **Note:** The length of the array property is specified by *[m]*, where m is the number of supported elements (according to the BACnet specification). The BACnet array properties that do not have a length specified by the BACnet standard is designated by *[N]*. This means the length of the property array depends on the BACnet device.

⬤ *For more information, refer to **Address Descriptions**, **Addressing Examples**.*

Property Mnemonic	BACnet Data Type	OPC Data Type	Access	COV
AckedTransitions[1] .ToFault .ToNormal .ToOffNormal	BACnetEventTransitionBits	**Word**, Short Boolean Boolean Boolean	Read Only	No
ActiveText	CharacterString	**String**	Read Only	No
AlarmValue	**BACnetBinaryPV**[2]	**Boolean**	Read/Write	No
ChangeOfStateCount	Unsigned Integer	**DWord**, Long	Read/Write	No
ChangeOfStateTime	BACnetDateTime	**String**	Read Only[3]	No
Description	CharacterString	**String**	Read Only	No
ElapsedActiveTime	Unsigned Integer	**DWord**, Long	Read Only	No
EventEnable[1] .ToFault .ToNormal .ToOffNormal	BACnetEventTransitionBits	**Word**, Short Boolean Boolean Boolean	Read/Write	No
EventState	**BACnetEventState**[2]	**DWord**, Long	Read Only	No
EventTimeStamps[3]	Array of TimeStamp	**String**	Read Only[3]	No
InactiveText	CharacterString	**String**	Read Only	No
MinimumOffTime	Unsigned Integer	**DWord**, Long	Read Only	No
MinimumOnTime	Unsigned Integer	**DWord**, Long	Read Only	No
NotificationClass	Unsigned Integer	**DWord**, Long	Read Only	No
NotifyType	**BACnetNotifyType**[2]	**DWord**, Long	Read Only	No
ObjectIdentifier[4] .ObjectInstance .ObjectType	BACnetObjectIdentifier	**DWord**, Long **DWord**, Long **DWord**, Long	Read Only[3]	No
ObjectName	CharacterString	**String**	Read Only	No
ObjectType	**BACnetObjectType**[2]	**DWord**, Long	Read Only[3]	No

Property Mnemonic	BACnet Data Type	OPC Data Type	Access	COV
OutOfService	Boolean	**Boolean**	Read/Write	No
PresentValue	**BACnetBinaryPV**[2]	**Boolean**	Read/Write	Yes
PriorityArray[16]	Array of BACnetPriorityArray	**String**	Read Only[3]	No
ProfileName	CharacterString	**String**	Read Only	No
Reliability	**BACnetReliability**[2]	**DWord**, Long	Read Only	No
RelinquishDefault	**BACnetBinaryPV**[2]	**Boolean**	Read/Write	No
StatusFlags[1] .Fault .InAlarm .OutOfService .Overridden	BACnetStatusFlags	**Word**, Short Boolean Boolean Boolean Boolean	Read Only	Yes
TimeDelay	Unsigned Integer	**DWord**, Long	Read/Write	No
TimeOfActiveTimeReset	BACnetDateTime	**String**	Read Only[3]	No
TimeOfStateCountReset	BACnetDateTime	**String**	Read Only[3]	No

⬤ **Notes**:

1. Bit string types may be viewed as a packed Word value, whose actual number of meaningful bits depends on the specific property. They may also be viewed as individual bits using optional sub-property fields.

2. Properties with enumerated BACnet Data Types are expressed as integer values. For standard interpretations, refer to **Enumerated Data Types**.

3. May not be made writable.

4. Object Identifier may be viewed as a packed DWord value (high 10 bits are the object type and low 22 bits are the object instance) or as individual tags for object type and instance using optional sub-property fields.

Priority Array Elements

Priority Array elements may be "NULL" or the numerical command value currently in effect. The array element index may range from 1 to 16, inclusive.

Calendar

The following table describes the object's supported properties. The Access column specifies the default access permission for tags. To accommodate non-standard implementations of BACnet, tags may be given Read/Write access unless noted otherwise.

⬤ **Notes:**

1. The length of the array property is specified by *[m]*, where m is the number of supported elements (according to the BACnet specification). The BACnet array properties that do not have a length specified by the BACnet standard is designated by *[N]*. This means the length of the property array depends on the BACnet device.

2. Implicit Change of Value (COV) notifications are not supported for this object. *For more information on devices with explicit COV support, refer to **COV Notifications**.*

● *For more information, refer to **Address Descriptions**, **Addressing Examples**, **COV Notifications**.*

Property Mnemonic	BACnet Data Type	OPC Data Type	Access
DateList[N]	CalendarEntry	**String**	Read Only[1]
DateList[2]	List of CalendarEntry	**String**	Read/Write
Description	CharacterString	**String**	Read Only
ObjectIdentifier[3] .ObjectInstance .ObjectType	BACnetObjectIdentifier	**DWord**, Long **DWord**, Long **DWord**, Long	Read Only[1]
ObjectName	CharacterString	**String**	Read Only
ObjectType	**BACnetObjectType**[4]	**DWord**, Long	Read Only[1]
PresentValue	Boolean	**Boolean**	Read/Write
ProfileName	CharacterString	**String**	Read Only

● **Notes**:

1. May not be made writable.

2. When without an array specifier, the DateList property represents the entire DateList as a formatted string. When with an array specifier, the DateList property is deprecated and included for legacy server project support. New projects should use the DateList property without an array specifier. *For more information, refer to **DateList String Format**.*

3. Object Identifier may be viewed as a packed DWord value (high 10 bits are the object type and low 22 bits are the object instance) or as individual tags for object type and instance using optional sub-property fields.

4. Properties with enumerated BACnet Data Types are expressed as integer values. For standard interpretations, refer to **Enumerated Data Types**.

Priority Array Elements

Priority Array elements may be "NULL" or the numerical command value currently in effect. The array element index may range from 1 to 16, inclusive.

Command

The following table describes the object's supported properties. The Access column specifies the default access permission for tags. To accommodate non-standard implementations of BACnet, tags may be given Read/Write access unless noted otherwise.

● **Notes**:

1. The length of the array property is specified by *[m]*, where m is the number of supported elements (according to the BACnet specification). The BACnet array properties that do not have a length specified by the BACnet standard is designated by *[N]*. This means the length of the property array

depends on the BACnet device. *For more information, refer to **Addressing Examples**.*

2. Implicit Change of Value (COV) notifications are not supported for this object. *For more information on devices with explicit COV support, refer to **COV Notifications**.*

*For more information, refer to **Address Descriptions**, **Addressing Examples**, **COV Notifications**.*

Property Mnemonic	BACnet Data Type	OPC Data Type	Access
ActionText[N]	Array of CharacterString	**String**	Read/Write
AllWritesSuccessful	Boolean	**Boolean**	Read Only
Description	CharacterString	**String**	Read Only
InProcess	Boolean	**Boolean**	Read Only
ObjectIdentifier[1] .ObjectInstance .ObjectType	BACnetObjectIdentifier	**DWord**, Long **DWord**, Long **DWord**, Long	Read Only[2]
ObjectName	CharacterString	**String**	Read Only
ObjectType	**BACnetObjectType**[3]	**DWord**, Long	Read Only[2]
PresentValue	Unsigned Integer	**DWord**, Long	Read/Write
ProfileName	CharacterString	**String**	Read Only

Notes:

1. Object Identifier may be viewed as a packed DWord value (high 10 bits are the object type and low 22 bits are the object instance) or as individual tags for object type and instance using optional sub-property fields.

2. May not be made writable.

3. Properties with enumerated BACnet Data Types are expressed as integer values. For standard interpretations, refer to **Enumerated Data Types**.

Priority Array Elements

Priority Array elements may be "NULL" or the numerical command value currently in effect. The array element index may range from 1 to 16, inclusive.

Device

The following table describes the object's supported properties. The Access column specifies the default access permission for tags. To accommodate non-standard implementations of BACnet, tags may be given Read/Write access unless noted otherwise.

Notes:

1. The length of the array property is specified by *[m]*, where m is the number of supported elements (according to the BACnet specification). The BACnet array properties that do not have a length specified by the BACnet standard is designated by *[N]*. This means the length of the property array depends on the BACnet device. *For more information, refer to **Addressing Examples**.*

2. Implicit Change of Value (COV) notifications are not supported for this object. *For more information on devices with explicit COV support, refer to* **COV Notifications**.

● *For more information, refer to* **Address Descriptions**, **Addressing Examples**, **COV Notifications**.

Property Mnemonic	BACnet Data Type	OPC Data Type	Access
APDUSegmentTimeout	Unsigned Integer	**DWord**, Long	Read/Write
APDUTimeout	Unsigned Integer	**DWord**, Long	Read/Write
ApplicationSoftwareVersion	CharacterString	**String**	Read Only
BackupFailureTimeout	Unsigned Integer	**Word**, Short	Read Only
ConfigurationFiles[N]	Array of ObjectIdentifier	**DWord**, Long	Read Only
DatabaseRevision	Unsigned Integer	**DWord**, Long	Read Only
DaylightSavingsStatus	Boolean	**Boolean**	Read Only
Description	CharacterString	**String**	Read Only
FirmwareRevision	CharacterString	**String**	Read Only
LastRestoreTime	BACnetDateTime	**String**	Read Only[1]
LocalDate	Date	**String**	Read Only[1]
LocalTime	Time	**String**	Read Only[1]
Location	CharacterString	**String**	Read Only
MaxAPDULengthAccepted	Unsigned Integer	**DWord**, Long	Read/Write
MaxInfoFrames	Unsigned Integer	**DWord**, Long	Read Only
MaxMaster	Unsigned Integer	**DWord**, Long	Read Only
MaxSegmentsAccepted	Unsigned Integer	**DWord**, Long	Read Only
ModelName	CharacterString	**String**	Read Only
NumberOfAPDURetries	Unsigned Integer	**DWord**, Long	Read/Write
ObjectIdentifier[2] .ObjectInstance .ObjectType	BACnetObjectIdentifier	**DWord**, Long **DWord**, Long **DWord**, Long	Read Only[1]
ObjectList[N]	Array of ObjectIdentifier	**DWord**, Long	Read Only[1]
ObjectName	CharacterString	**String**	Read Only
ObjectType	**BACnetObjectType**[3]	**DWord**, Long	Read Only[1]
ProfileName	CharacterString	**String**	Read Only
ProtocolRevision	Unsigned Integer	**DWord**, Long	Read Only
ProtocolVersion	Unsigned Integer	**DWord**, Long	Read Only
SegmentationSupported	**BACnetSegmentation**[3]	**DWord**, Long	Read Only
SystemStatus	**BACnetDeviceStatus**[3]	**DWord**, Long	Read/Write
UTCOffset	INTEGER	**Long**, Word	Read Only
VendorIdentifier	Unsigned Integer	**Word**, Short	Read Only
VendorName	CharacterString	**String**	Read Only

● **Notes**:

1. May not be made writable.

2. Object Identifier may be viewed as a packed DWord value (high 10 bits are the object type and low 22 bits are the object instance) or as individual tags for object type and instance using optional sub-property fields.

3. Properties with enumerated BACnet Data Types are expressed as integer values. For standard interpretations, refer to **Enumerated Data Types**.

Priority Array Elements

Priority Array elements may be "NULL" or the numerical command value currently in effect. The array element index may range from 1 to 16, inclusive.

Event Enrollment

The following table describes the object's supported properties. The Access column specifies the default access permission for tags. To accommodate non-standard implementations of BACnet, tags may be given Read/Write access unless noted otherwise.

Notes:

1. The length of the array property is specified by [m], where m is the number of supported elements (according to the BACnet specification). The BACnet array properties that do not have a length specified by the BACnet standard is designated by [N]. This means the length of the property array depends on the BACnet device. *For more information, refer to **Addressing Examples**.*

2. Implicit Change of Value (COV) notifications are not supported for this object. *For more information on devices with explicit COV support, refer to **COV Notifications**.*

*For more information, refer to **Address Descriptions**, **Addressing Examples**, **COV Notifications**.*

Property Mnemonic	BACnet Data Type	OPC Data Type	Access
AckedTransitions[1]	BACnetEventTransitionBits	**Word**, Short	Read Only
.ToFault		Boolean	
.ToNormal		Boolean	
.ToOffNormal		Boolean	
Description	CharacterString	**String**	Read Only
EventEnable[1]	BACnetEventTransitionBits	**Word**, Short	Read/Write
.ToFault		Boolean	
.ToNormal		Boolean	
.ToOffNormal		Boolean	
EventState	**BACnetEventState**[2]	**DWord**, Long	Read Only
EventTimeStamps[3]	Array of TimeStamp	**String**	Read Only[3]
EventType	**BACnetEventType**[2]	**DWord**, Long	Read Only
IssueConfirmedNotifications	Boolean	**Boolean**	Read/Write
NotificationClass	Unsigned Integer	**Word**, Short	Read Only
NotifyType	**BACnetNotifyType**[2]	**DWord**, Long	Read Only

Property Mnemonic	BACnet Data Type	OPC Data Type	Access
ObjectIdentifier[4] .ObjectInstance .ObjectType	BACnetObjectIdentifier	**DWord**, Long **DWord**, Long **DWord**, Long	Read Only[3]
ObjectName	CharacterString	**String**	Read Only
ObjectType	**BACnetObjectType**[2]	**DWord**, Long	Read Only[3]
Priority	Unsigned Integer	**DWord**, Long	Read Only
ProcessIdentifier	Unsigned Integer	**DWord**, Long	Read/Write
ProfileName	CharacterString	**String**	Read Only

Notes:

1. Bit string types may be viewed as a packed Word value, whose actual number of meaningful bits depends on the specific property. They may also be viewed as individual bits using optional sub-property fields.

2. Properties with enumerated BACnet Data Types are expressed as integer values. For standard interpretations, refer to **Enumerated Data Types**.

3. May not be made writable.

4. Object Identifier may be viewed as a packed DWord value (high 10 bits are the object type and low 22 bits are the object instance) or as individual tags for object type and instance using optional sub-property fields.

Priority Array Elements

Priority Array elements may be "NULL" or the numerical command value currently in effect. The array element index may range from 1 to 16, inclusive.

File

The following table describes the object's supported properties. The Access column specifies the default access permission for tags. To accommodate non-standard implementations of BACnet, tags may be given Read/Write access unless noted otherwise.

Notes:

1. The length of the array property is specified by *[m]*, where m is the number of supported elements (according to the BACnet specification). The BACnet array properties that do not have a length specified by the BACnet standard is designated by *[N]*. This means the length of the property array depends on the BACnet device. *For more information, refer to **Addressing Examples**.*

2. Implicit Change of Value (COV) notifications are not supported for this object. *For more information on devices with explicit COV support, refer to **COV Notifications**.*

*For more information, refer to **Address Descriptions**, **Addressing Examples**, **COV Notifications**.*

Property Mnemonic	BACnet Data Type	OPC Data Type	Access
Archive	Boolean	**Boolean**	Read Only

Property Mnemonic	BACnet Data Type	OPC Data Type	Access
Description	CharacterString	**String**	Read Only
FileAccessMethod	**BACnetFileAccessMethod**[1]	**DWord**, Long	Read Only
FileSize	Unsigned Integer	**DWord**, Long	Read Only
FileType	CharacterString	**String**	Read Only
ModificationDate	BACnetDateTime	**String**	Read Only[2]
ObjectIdentifier[3] .ObjectInstance .ObjectType	BACnetObjectIdentifier	**DWord**, Long **DWord**, Long **DWord**, Long	Read Only[2]
ObjectName	CharacterString	**String**	Read Only
ObjectType	**BACnetObjectType**[1]	**DWord**, Long	Read Only[2]
ProfileName	CharacterString	**String**	Read Only
ReadOnly	Boolean	**Boolean**	Read Only
RecordCount	Unsigned Integer	**DWord**, Long	Read/Write

🔵 **Notes**:

1. Properties with enumerated BACnet Data Types are expressed as integer values. For standard interpretations, refer to **Enumerated Data Types**.

2. May not be made writable.

3. Object Identifier may be viewed as a packed DWord value (high 10 bits are the object type and low 22 bits are the object instance) or as individual tags for object type and instance using optional sub-property fields.

Priority Array Elements

Priority Array elements may be "NULL" or the numerical command value currently in effect. The array element index may range from 1 to 16, inclusive.

Group

The following table describes the object's supported properties. The Access column specifies the default access permission for tags. To accommodate non-standard implementations of BACnet, tags may be given Read/Write access unless noted otherwise.

🔵 **Notes:**

1. The length of the array property is specified by *[m]*, where m is the number of supported elements (according to the BACnet specification). The BACnet array properties that do not have a length specified by the BACnet standard is designated by *[N]*. This means the length of the property array depends on the BACnet device. *For more information, refer to **Addressing Examples**.*

2. Implicit Change of Value (COV) notifications are not supported for this object. *For more information on devices with explicit COV support, refer to **COV Notifications**.*

🔵 *For more information, refer to **Address Descriptions**, **Addressing Examples**, **COV Notifications**.*

Property Mnemonic	BACnet Data Type	OPC Data Type	Access
Description	CharacterString	**String**	Read Only
ObjectIdentifier[1]	BACnetObjectIdentifier	**DWord**, Long	Read Only[2]
.ObjectInstance		**DWord**, Long	
.ObjectType		**DWord**, Long	
ObjectName	CharacterString	**String**	Read Only
ObjectType	**BACnetObjectType**[3]	**DWord**, Long	Read Only[2]
ProfileName	CharacterString	**String**	Read Only

Notes:

1. Object Identifier may be viewed as a packed DWord value (high 10 bits are the object type and low 22 bits are the object instance) or as individual tags for object type and instance using optional sub-property fields.

2. May not be made writable.

3. Properties with enumerated BACnet Data Types are expressed as integer values. For standard interpretations, refer to **Enumerated Data Types**.

Priority Array Elements

Priority Array elements may be "NULL" or the numerical command value currently in effect. The array element index may range from 1 to 16, inclusive.

Life Safety Point

The following table describes the object's supported properties. The Access column specifies the default access permission for tags. To accommodate non-standard implementations of BACnet, tags may be given Read/Write access unless noted otherwise. The COV column specifies whether the driver considers the property to have implicit Change Of Value (COV) notification capability; that is, whether the BACnet specification requires the property to support COV. For some properties, COV support depends on implementation. The "COV" modifier must be added to the tag's address for use. *For more information, refer to COV Settings*.

Note: The length of the array property is specified by *[m]*, where m is the number of supported elements (according to the BACnet specification). The BACnet array properties that do not have a length specified by the BACnet standard is designated by *[N]*. This means the length of the property array depends on the BACnet device. *For more information, refer to Addressing Examples*.

For more information, refer to Address Descriptions, Addressing Examples, COV Notifications.

Property Mnemonic	BACnet Data Type	OPC Data Type	Access	COV
AcceptedModes	List of **BACnetLifeSafetyMode**	**DWord**, Long	Read Only[1]	No
AckedTransitions[2]	BACnetEventTansitionBits	**Word**, Short	Read Only	No
.ToFault		Boolean		
.ToNormal		Boolean		
.ToOffNormal		Boolean		
AlarmValues	List of **BACnetLifeSafetyState**	**DWord**, Long	Read Only[1]	No

Property Mnemonic	BACnet Data Type	OPC Data Type	Access	COV
Description	CharacterString	**String**	Read Only	No
DeviceType	CharacterString	**String**	Read Only	No
DirectReading	REAL	**Float**	Read Only	No
EventEnable[2] .ToFault .ToNormal .ToOffNormal	BACnetEventTransitionBits	**Word**, Short Boolean Boolean Boolean	Read/Write	No
EventState	**BACnetEventState**[3]	**DWord**, Long	Read Only	No
EventTimeStamps[3]	Array of TimeStamp	**String**	Read Only[1]	No
FaultValues	List of **BACnetLifeSafetyState**	**DWord**, Long	Read Only[1]	No
LifeSafetyAlarmValues	List of **BACnetLifeSafetyState**	**DWord**, Long	Read Only[1]	No
MaintenanceRequired	**BACnetMaintenance**[3]	**DWord**, Long	Read/Write	No
Mode	**BACnetLifeSafetyMode**[3]	**DWord**, Long	Read/Write	No
NotificationClass	Unsigned Integer	**DWord**, Long	Read Only	No
NotifyType	**BACnetNotifyType**[3]	**DWord**, Long	Read Only	No
ObjectIdentifier[4] .ObjectInstance .ObjectType	BACnetObjectIdentifier	**DWord**, Long **DWord**, Long **DWord**, Long	Read Only[1]	No
ObjectName	CharacterString	**String**	Read Only	No
ObjectType	**BACnetObjectType**[3]	**DWord**, Long	Read Only[1]	No
OperationExpected	**BACnetLifeSafetyOperation**[3]	**DWord**, Long	Read Only	No
OutOfService	Boolean	**Boolean**	Read/Write	No
PresentValue	**BACnetLifeSafetyState**[3]	**DWord**, Long	Read/Write	Yes
ProfileName	CharacterString	**String**	Read Only	No
Reliability	**BACnetReliability**[3]	**DWord**, Long	Read Only	No
Setting	Unsigned Integer	**Byte**, Char	Read/Write	No
Silenced	**BACnetSilencedState**[3]	**DWord**, Long	Read Only	No
StatusFlags[2] .Fault .InAlarm .OutOfService .Overridden	BACnetStatusFlags	**Word**, Short Boolean Boolean Boolean Boolean	Read Only	Yes
TimeDelay	Unsigned Integer	**DWord**, Long	Read/Write	No
TrackingValue	**BACnetLifeSafetyState**[3]	**DWord**, Long	Read Only	No
Units	**BACnetEngineeringUnits**[3]	**DWord**, Long	Read Only	No

Notes:

1. May not be made writable.

2. Bit string types may be viewed as a packed Word value, whose actual number of meaningful bits depends on the specific property. They may also be viewed as individual bits using optional sub-property fields.

3. Properties with enumerated BACnet Data Types are expressed as integer values. For standard interpretations, refer to **Enumerated Data Types**.

4. Object Identifier may be viewed as a packed DWord value (high 10 bits are the object type and low 22 bits are the object instance) or as individual tags for object type and instance using optional sub-property fields.

Priority Array Elements

Priority Array elements may be "NULL" or the numerical command value currently in effect. The array element index may range from 1 to 16, inclusive.

Life Safety Zone

The following table describes the object's supported properties. The Access column specifies the default access permission for tags. To accommodate non-standard implementations of BACnet, tags may be given Read/Write access unless noted otherwise. The COV column specifies whether the driver considers the property to have implicit Change Of Value (COV) notification capability; that is, whether the BACnet specification requires the property to support COV. For some properties, COV support depends on implementation. The "COV" modifier must be added to the tag's address for use. *For more information, refer to **COV Settings**.*

 Note: The length of the array property is specified by *[m]*, where m is the number of supported elements (according to the BACnet specification). The BACnet array properties that do not have a length specified by the BACnet standard is designated by *[N]*. This means the length of the property array depends on the BACnet device. *For more information, refer to **Addressing Examples**.*

 *For more information, refer to **Address Descriptions**, **Addressing Examples**, **COV Notifications**.*

Property Mnemonic	BACnet Data Type	OPC Data Type	Access	COV
AcceptedModes	List of **BACnetLifeSafetyMode**	**DWord**, Long	Read Only[1]	No
AckedTransitions[2] .ToFault .ToNormal .ToOffNormal	BACnetEventTransitionBits	**Word**, Short Boolean Boolean Boolean	Read Only	No
AlarmValues	List of **BACnetLifeSafetyState**	**DWord**, Long	Read Only[1]	No
Description	CharacterString	String	Read Only	No
DeviceType	CharacterString	String	Read Only	No
EventEnable[2] .ToFault .ToNormal .ToOffNormal	BACnetEventTransitionBits	**Word**, Short Boolean Boolean Boolean	Read/Write	No
EventState	**BACnetEventState**[3]	**DWord**, Long	Read Only	No

Property Mnemonic	BACnet Data Type	OPC Data Type	Access	COV
EventTimeStamps[3]	Array of TimeStamp	**String**	Read Only[1]	No
FaultValues	List of **BACnetLifeSafetyState**	**DWord**, Long	Read Only[1]	No
LifeSafetyAlarmValues	List of **BACnetLifeSafetyState**	**DWord**, Long	Read Only[1]	No
MaintenanceRequired	Boolean	Boolean	Read/Write	No
Mode	**BACnetLifeSafetyMode**[3]	**DWord**, Long	Read/Write	No
NotificationClass	Unsigned Integer	**DWord**, Long	Read Only	No
NotifyType	**BACnetNotifyType**[3]	**DWord**, Long	Read Only	No
ObjectIdentifier[4] .ObjectInstance .ObjectType	BACnetObjectIdentier	**DWord**, Long **DWord**, Long **DWord**, Long	Read Only[1]	No
ObjectName	CharacterString	String	Read Only	No
ObjectType	**BACnetObjectType**[3]	**DWord**, Long	Read Only[1]	No
OperationExpected	**BACnetLifeSafetyOperation**[3]	**DWord**, Long	Read Only	No
OutOfService	Boolean	Boolean	Read/Write	No
PresentValue	**BACnetLifeSafetyState**[3]	**DWord**, Long	Read/Write	Yes
ProfileName	CharacterString	String	Read Only	No
Reliability	**BACnetReliability**[3]	**DWord**, Long	Read Only	No
Silenced	**BACnetSilencedState**[3]	**DWord**, Long	Read Only	No
StatusFlags[2] .Fault .InAlarm .OutOfService .Overridden	BACnetStatusFlags	**Word**, Short Boolean Boolean Boolean Boolean	Read Only	Yes
TimeDelay	Unsigned Integer	**DWord**, Long	Read/Write	No
TrackingValue	**BACnetLifeSafetyState**[3]	**DWord**, Long	Read Only	No

Notes:

1. May not be made writable.

2. Bit string types may be viewed as a packed Word value, whose actual number of meaningful bits depends on the specific property. They may also be viewed as individual bits using optional sub-property fields.

3. Properties with enumerated BACnet Data Types are expressed as integer values. For standard interpretations, refer to **Enumerated Data Types**.

4. Object Identifier may be viewed as a packed DWord value (high 10 bits are the object type and low 22 bits are the object instance) or as individual tags for object type and instance using optional sub-property fields.

Priority Array Elements

Priority Array elements may be "NULL" or the numerical command value currently in effect. The array element index may range from 1 to 16, inclusive.

Loop

The following table describes the object's supported properties. The Access column specifies the default access permission for tags. To accommodate non-standard implementations of BACnet, tags may be given Read/Write access unless noted otherwise. The COV column specifies whether the driver considers the property to have implicit Change Of Value (COV) notification capability; that is, whether the BACnet specification requires the property to support COV. For some properties, COV support depends on implementation. The "COV" modifier must be added to the tag's address for use. *For more information, refer to COV Settings*.

● **Note**: The length of the array property is specified by *[m]*, where m is the number of supported elements (according to the BACnet specification). The BACnet array properties that do not have a length specified by the BACnet standard is designated by *[N]*. This means the length of the property array depends on the BACnet device.

● *For more information, refer to Address Descriptions, Addressing Examples, COV Notifications*.

Property Mnemonic	BACnet Data Type	OPC Data Type	Access	COV
AckedTransitions[1] .ToFault .ToNormal .ToOffNormal	BACnetEventTransitionBits	**Word**, Short Boolean Boolean Boolean	Read Only	No
Action	**BACnetAction**[2]	**DWord**, Long	Read Only	No
Bias	REAL	**Float**	Read/Write	No
ControlledVariableUnits	**BACnetEngineeringUnits**[2]	**DWord**, Long	Read Only	No
ControlledVariableValue	REAL	**Float**	Read Only	Yes
COVIncrement	REAL	**Float**	Read/Write	No
DerivativeConstant	REAL	**Float**	Read/Write	No
DerivativeConstantUnits	**BACnetEngineeringUnits**[2]	**DWord**, Long	Read Only	No
Description	CharacterString	**String**	Read Only	No
ErrorLimit	REAL	**Float**	Read/Write	No
EventEnable[1] .ToFault .ToNormal .ToOffNormal	BACnetEventTransitionBits	**Word**, Short Boolean Boolean Boolean	Read/Write	No
EventState	**BACnetEventState**[2]	**DWord**, Long	Read Only	No
EventTimeStamps[3]	Array of TimeStamp	**String**	Read Only[3]	No
IntegralConstant	REAL	**Float**	Read/Write	No
IntegralConstantUnits	**BACnetEngineeringUnits**[2]	**DWord**, Long	Read Only	No
MaximumOutput	REAL	**Float**	Read Only	No
MinimumOutput	REAL	**Float**	Read Only	No
NotificationClass	Unsigned Integer	**DWord**, Long	Read Only	No
NotifyType	BACnetNotifyType	**DWord**, Long	Read Only	No
ObjectIdentifier[4]	BACnetObjectIdentifier	**DWord**, Long	Read Only[3]	No

Property Mnemonic	BACnet Data Type	OPC Data Type	Access	COV
.ObjectInstance .ObjectType		**DWord**, Long **DWord**, Long		
ObjectName	CharacterString	**String**	Read Only	No
ObjectType	**BACnetObjectType**[2]	**DWord**, Long	Read Only[3]	No
OutOfService	Boolean	**Boolean**	Read/Write	No
OutputUnits	**BACnetEngineeringUnits**[2]	**DWord**, Long	Read Only	No
PresentValue	REAL	**Float**	Read Only	Yes
PriorityForWriting	Unsigned Integer	**DWord**, Long	Read/Write	No
ProfileName	CharacterString	**String**	Read Only	No
ProportionalConstant	REAL	**Float**	Read/Write	No
ProportionalConstantUnits	**BACnetEngineeringUnits**[2]	**DWord**, Long	Read Only	No
Reliability	**BACnetReliability**[2]	**DWord**, Long	Read Only	No
Setpoint	REAL	**Float**	Read/Write	Yes
StatusFlags[1] .Fault .InAlarm .OutOfService .Overridden	BACnetStatusFlags	**Word**, Short Boolean Boolean Boolean Boolean	Read Only	Yes
TimeDelay	Unsigned Integer	**DWord**, Long	Read/Write	No
UpdateInterval	Unsigned Integer	**DWord**, Long	Read/Write	No

Notes:

1. Bit string types may be viewed as a packed Word value, whose actual number of meaningful bits depends on the specific property. They may also be viewed as individual bits using optional sub-property fields.

2. Properties with enumerated BACnet Data Types are expressed as integer values. For standard interpretations, refer to **Enumerated Data Types**.

3. May not be made writable.

4. Object Identifier may be viewed as a packed DWord value (high 10 bits are the object type and low 22 bits are the object instance) or as individual tags for object type and instance using optional sub-property fields.

Priority Array Elements

Priority Array elements may be "NULL" or the numerical command value currently in effect. The array element index may range from 1 to 16, inclusive.

Multi-State Input

The following table describes the object's supported properties. The Access column specifies the default access permission for tags. To accommodate non-standard implementations of BACnet, tags may be given

Read/Write access unless noted otherwise. The COV column specifies whether the driver considers the property to have implicit Change Of Value (COV) notification capability; that is, whether the BACnet specification requires the property to support COV. For some properties, COV support depends on implementation. The "COV" modifier must be added to the tag's address for use. *For more information, refer to* **COV Settings**.

Note: The length of the array property is specified by *[m]*, where m is the number of supported elements (according to the BACnet specification). The BACnet array properties that do not have a length specified by the BACnet standard is designated by *[N]*. This means the length of the property array depends on the BACnet device.

For more information, refer to **Address Descriptions**, **Addressing Examples**, **COV Notifications**.

Property Mnemonic	BACnet Data Type	OPC Data Type	Access	COV
AckedTransitions[1] .ToFault .ToNormal .ToOffNormal	BACnetEventTransitionBits	**Word**, Short Boolean Boolean Boolean	Read Only	No
AlarmValues	List of Unsigned Integers	**DWord**, Long	Read Only[2]	No
Description	CharacterString	**String**	Read Only	No
DeviceType	CharacterString	**String**	Read Only	No
EventEnable[1] .ToFault .ToNormal .ToOffNormal	BACnetEventTransitionBits	**Word**, Short Boolean Boolean Boolean	Read/Write	No
EventState	**BACnetEventState**[3]	**DWord**, Long	Read Only	No
EventTimeStamps[3]	Array of TimeStamp	**String**	Read Only[2]	No
FaultValues	List of Unsigned Integers	**DWord**, Long	Read Only[2]	No
NotificationClass	Unsigned Integer	**DWord**, Long	Read Only	No
NotifyType	**BACnetNotifyType**[3]	**DWord**, Long	Read Only	No
NumberOfStates	Unsigned Integer	**DWord**, Long	Read Only	No
ObjectIdentifier[4] .ObjectInstance .ObjectType	BACnetObjectIdentifier	**DWord**, Long **DWord**, Long **DWord**, Long	Read Only[2]	No
ObjectName	CharacterString	**String**	Read Only	No
ObjectType	**BACnetObjectType**[3]	**DWord**, Long	Read Only[2]	No
OutOfService	Boolean	**Boolean**	Read/Write	No
PresentValue	Unsigned Integer	**DWord**, Long	Read/Write	Yes
ProfileName	CharacterString	**String**	Read Only	No
Reliability	**BACnetReliability**[3]	**DWord**, Long	Read Only	No
StateText[N]	Array of CharacterString	**String**	Read/Write	No
StatusFlags[1] .Fault	BACnetStatusFlags	**Word**, Short Boolean	Read Only	Yes

Property Mnemonic	BACnet Data Type	OPC Data Type	Access	COV
.InAlarm .OutOfService .Overridden		Boolean Boolean Boolean		
TimeDelay	Unsigned Integer	**DWord**, Long	Read/Write	No

⬤ **Notes**:

1. Bit string types may be viewed as a packed Word value, whose actual number of meaningful bits depends on the specific property. They may also be viewed as individual bits using optional sub-property fields.

2. May not be made writable.

3. Properties with enumerated BACnet Data Types are expressed as integer values. For standard interpretations, refer to **Enumerated Data Types**.

4. Object Identifier may be viewed as a packed DWord value (high 10 bits are the object type and low 22 bits are the object instance) or as individual tags for object type and instance using optional sub-property fields.

Priority Array Elements

Priority Array elements may be "NULL" or the numerical command value currently in effect. The array element index may range from 1 to 16, inclusive.

Multi-State Output

The following table describes the object's supported properties. The Access column specifies the default access permission for tags. To accommodate non-standard implementations of BACnet, tags may be given Read/Write access unless noted otherwise. The COV column specifies whether the driver considers the property to have implicit Change Of Value (COV) notification capability; that is, whether the BACnet specification requires the property to support COV. For some properties, COV support depends on implementation. The "COV" modifier must be added to the tag's address for use. *For more information, refer to COV Settings*.

⬤ **Note**: The length of the array property is specified by *[m]*, where m is the number of supported elements (according to the BACnet specification). The BACnet array properties that do not have a length specified by the BACnet standard is designated by *[N]*. This means the length of the property array depends on the BACnet device.

⬤ *For more information, refer to Address Descriptions, Addressing Examples, COV Notifications.*

Property Mnemonic	BACnet Data Type	OPC Data Type	Access	COV
AckedTransitions[1] .ToFault .ToNormal .ToOffNormal	BACnetEventTransitionBits	**Word**, Short Boolean Boolean Boolean	Read Only	No
Description	CharacterString	**String**	Read Only	No
DeviceType	CharacterString	**String**	Read Only	No
EventEnable[1]	BACEventTransitionBits	**Word**, Short	Read/Write	No

Property Mnemonic	BACnet Data Type	OPC Data Type	Access	COV
.ToFault .ToNormal .ToOffNormal		Boolean Boolean Boolean		
EventState	**BACnetEventState**[2]	**DWord**, Long	Read Only	No
EventTimeStamps[3]	Array of TimeStamp	**String**	Read Only[3]	No
FeedbackValue	Unsigned Integer	**DWord**, Long	Read/Write	No
NotificationClass	Unsigned Integer	**DWord**, Long	Read Only	No
NotifyType	**BACnetNotifyType**[2]	**DWord**, Long	Read Only	No
NumberOfStates	Unsigned Integer	**DWord**, Long	Read Only	No
ObjectIdentifier[4] .ObjectInstance .ObjectType	BACnetObjectIdentifier	**DWord**, Long **DWord**, Long **DWord**, Long	Read Only[3]	No
ObjectName	CharacterString	**String**	Read Only	No
ObjectType	**BACnetObjectType**[2]	**DWord**, Long	Read Only[3]	No
OutOfService	Boolean	**Boolean**	Read/Write	No
PresentValue	Unsigned Integer	**DWord**, Long	Read/Write	Yes
PriorityArray[16]	Array of BACnetPriorityArray	**String**	Read Only[3]	No
ProfileName	CharacterString	**String**	Read Only	No
Reliability	**BACnetReliability**[2]	**DWord**, Long	Read Only	No
RelinquishDefault	Unsigned Integer	**DWord**, Long	Read/Write	No
StateText[N]	Array of CharacterString	**String**	Read/Write	No
StatusFlags[1] .Fault .InAlarm .OutOfService .Overridden	BACnetStatusFlags	**Word**, Short Boolean Boolean Boolean Boolean	Read Only	Yes
TimeDelay	Unsigned Integer	**DWord**, Long	Read/Write	No

🔹 **Notes**:

1. Bit string types may be viewed as a packed Word value, whose actual number of meaningful bits depends on the specific property. They may also be viewed as individual bits using optional sub-property fields.

2. Properties with enumerated BACnet Data Types are expressed as integer values. For standard interpretations, refer to **Enumerated Data Types**.

3. May not be made writable.

4. Object Identifier may be viewed as a packed DWord value (high 10 bits are the object type and low 22 bits are the object instance) or as individual tags for object type and instance using optional sub-property fields.

Priority Array Elements

Priority Array elements may be "NULL" or the numerical command value currently in effect. The array element index may range from 1 to 16, inclusive.

Multi-State Value

The following table describes the object's supported properties. The Access column specifies the default access permission for tags. To accommodate non-standard implementations of BACnet, tags may be given Read/Write access unless noted otherwise. The COV column specifies whether the driver considers the property to have implicit Change Of Value (COV) notification capability; that is, whether the BACnet specification requires the property to support COV. For some properties, COV support depends on implementation. The "COV" modifier must be added to the tag's address for use. *For more information, refer to COV Settings.*

● **Note**: The length of the array property is specified by *[m]*, where m is the number of supported elements (according to the BACnet specification). The BACnet array properties that do not have a length specified by the BACnet standard is designated by *[N]*. This means the length of the property array depends on the BACnet device.

● *For more information, refer to **Address Descriptions**, **Addressing Examples**, **COV Notifications**.*

Property Mnemonic	BACnet Data Type	OPC Data Type	Access	COV
AckedTransitions[1] .ToFault .ToNormal .ToOffNormal	BACnetEventTransitionBits	**Word**, Short Boolean Boolean Boolean	Read Only	No
AlarmValues	List of Unsigned Integers	**DWord**, Long	Read Only[2]	No
Description	CharacterString	**String**	Read Only	No
EventEnable[1] .ToFault .ToNormal .ToOffNormal	BACnetEventTransitionBits	**Word**, Short Boolean Boolean Boolean	Read/Write	No
EventState	**BACnetEventState**[3]	**DWord**, Long	Read Only	No
EventTimeStamps[3]	Array of TimeStamp	**String**	Read Only[2]	No
FaultValues	List of Unsigned Integers	**DWord**, Long	Read Only[2]	No
NotificationClass	Unsigned Integer	**DWord**, Long	Read Only	No
NotifyType	**BACnetNotifyType**[3]	**DWord**, Long	Read Only	No
NumberOfStates	Unsigned Integer	**DWord**, Long	Read Only	No
ObjectIdentifier[4] .ObjectInstance .ObjectType	BACnetObjectIdentifier	**DWord**, Long **DWord**, Long **DWord**, Long	Read Only[2]	No
ObjectName	CharacterString	**String**	Read Only	No
ObjectType	**BACnetObjectType**[3]	**DWord**, Long	Read Only[2]	No
OutOfService	Boolean	**Boolean**	Read/Write	No
PresentValue	Unsigned Integer	**DWord**, Long	Read/Write	Yes

Property Mnemonic	BACnet Data Type	OPC Data Type	Access	COV
PriorityArray[16]	Array of BACnetPriorityArray	**String**	Read Only[2]	No
ProfileName	CharacterString	**String**	Read Only	No
Reliability	**BACnetReliability**[3]	**DWord**, Long	Read Only	No
RelinquishDefault	Unsigned Integer	**DWord**, Long	Read/Write	No
StateText[N]	Array of CharacterString	**String**	Read/Write	No
StatusFlags[1] .Fault .InAlarm .OutOfService .Overridden	BACnetStatusFlags	**Word**, Short Boolean Boolean Boolean Boolean	Read Only	Yes
TimeDelay	Unsigned Integer	**DWord**, Long	Read/Write	No

⬤ **Notes**:

1. Bit string types may be viewed as a packed Word value, whose actual number of meaningful bits depends on the specific property. They may also be viewed as individual bits using optional sub-property fields.

2. May not be made writable.

3. Properties with enumerated BACnet Data Types are expressed as integer values. For standard interpretations, refer to **Enumerated Data Types**.

4. Object Identifier may be viewed as a packed DWord value (high 10 bits are the object type and low 22 bits are the object instance) or as individual tags for object type and instance using optional sub-property fields.

Priority Array Elements

Priority Array elements may be "NULL" or the numerical command value currently in effect. The array element index may range from 1 to 16, inclusive.

Notification Class

The following table describes the object's supported properties. The Access column specifies the default access permission for tags. To accommodate non-standard implementations of BACnet, tags may be given Read/Write access unless noted otherwise.

⬤ **Notes:**

1. The length of the array property is specified by *[m]*, where m is the number of supported elements (according to the BACnet specification). The BACnet array properties that do not have a length specified by the BACnet standard is designated by *[N]*. This means the length of the property array depends on the BACnet device.

2. Implicit Change of Value (COV) notifications are not supported for this object. *For more information on devices with explicit COV support, refer to **COV Notifications**.*

⬤ For more information, refer to **Address Descriptions**, **Addressing Examples**, **COV Notifications**.

Property Mnemonic	BACnet Data Type	OPC Data Type	Access
AckRequired[1] .ToFault .ToNormal .ToOffNormal	BACnetEvnetTransitionBits	**Word**, Short Boolean Boolean Boolean	Read Only
Description	CharacterString	String	Read Only
NotificationClass	Unsigned Integer	**DWord**, Long	Read Only
ObjectIdentifier[2] .ObjectInstance .ObjectType	BACnetObjectIdentifier	**DWord**, Long **DWord**, Long **DWord**, Long	Read Only[3]
ObjectName	CharacterString	String	Read Only
ObjectType	**BACnetObjectType**[4]	**DWord**, Long	Read Only[3]
Priority[3]	Array of Unsigned Integers	**DWord**, Long	Read Only[3]
ProfileName	CharacterString	String	Read Only

Notes:

1. Bit string types may be viewed as a packed Word value, whose actual number of meaningful bits depends on the specific property. They may also be viewed as individual bits using optional sub-property fields.

2. Object Identifier may be viewed as a packed DWord value (high 10 bits are the object type and low 22 bits are the object instance) or as individual tags for object type and instance using optional sub-property fields.

3. May not be made writable.

4. Properties with enumerated BACnet Data Types are expressed as integer values. For standard interpretations, refer to **Enumerated Data Types**.

Priority Array Elements

Priority Array elements may be "NULL" or the numerical command value currently in effect. The array element index may range from 1 to 16, inclusive.

Program

The following table describes the object's supported properties. The Access column specifies the default access permission for tags. To accommodate non-standard implementations of BACnet, tags may be given Read/Write access unless noted otherwise.

Notes:

1. The length of the array property is specified by *[m]*, where m is the number of supported elements (according to the BACnet specification). The BACnet array properties that do not have a length specified by the BACnet standard is designated by *[N]*. This means the length of the property array depends on the BACnet device. *For more information, refer to **Addressing Examples**.*

2. Implicit Change of Value (COV) notifications are not supported for this object. *For more information on devices with explicit COV support, refer to* **COV Notifications**.

For more information, refer to **Address Descriptions**, **Addressing Examples**, **COV Notifications**.

Property Mnemonic	BACnet Data Type	OPC Data Type	Access
Description	CharacterString	**String**	Read Only
DescriptionOfHalt	CharacterString	**String**	Read Only
InstanceOf	CharacterString	**String**	Read Only
ObjectIdentifier[1] .ObjectInstance .ObjectType	BACnetObjectIdentifier	**DWord**, Long **DWord**, Long **DWord**, Long	Read Only[2]
ObjectName	CharacterString	**String**	Read Only
ObjectType	**BACnetObjectType**[3]	**DWord**, Long	Read Only[2]
OutOfService	Boolean	**Boolean**	Read/Write
ProfileName	CharacterString	**String**	Read Only
ProgramChange	**BACnetProgramRequest**[3]	**DWord**, Long	Read Only
ProgramLocation	CharacterString	**String**	Read Only
ProgramState	**BACnetProgramState**[3]	**DWord**, Long	Read Only
ReasonForHalt	**BACnetProgramError**[3]	**DWord**, Long	Read Only
Reliability	**BACnetReliability**[3]	**DWord**, Long	Read Only
StatusFlags[4] .Fault .InAlarm .OutOfService .Overridden	BACnetStatusFlags	**Word**, Short Boolean Boolean Boolean Boolean	Read Only

Notes:

1. Object Identifier may be viewed as a packed DWord value (high 10 bits are the object type and low 22 bits are the object instance) or as individual tags for object type and instance using optional sub-property fields.

2. May not be made writable.

3. Properties with enumerated BACnet Data Types are expressed as integer values. For standard interpretations, refer to **Enumerated Data Types**.

4. Bit string types may be viewed as a packed Word value, whose actual number of meaningful bits depends on the specific property. They may also be viewed as individual bits using optional sub-property fields.

Priority Array Elements

Priority Array elements may be "NULL" or the numerical command value currently in effect. The array element index may range from 1 to 16, inclusive.

Schedule

The following table describes the object's supported properties. The Access column specifies the default access permission for tags. To accommodate non-standard implementations of BACnet, tags may be given Read/Write access unless noted otherwise. The COV column specifies whether the driver considers the property to have implicit Change Of Value (COV) notification capability; that is, whether the BACnet specification requires the property to support COV. For some properties, COV support depends on implementation. The "COV" modifier must be added to the tag's address for use. *For more information, refer to **COV Settings**.*

Note: The length of the array property is specified by *[m]*, where m is the number of supported elements (according to the BACnet specification). The BACnet array properties that do not have a length specified by the BACnet standard is designated by *[N]*. This means the length of the property array depends on the BACnet device. *For more information, refer to **Addressing Examples**.*

*For more information, refer to **Address Descriptions**, **Addressing Examples**, **COV Notifications**.*

Property Mnemonic	BACnet Data Type	OPC Data Type	Access	COV
Description	CharacterString	**String**	Read Only	No
EffectivePeriod	BACnetDateRange	**String**	Read Only[1]	No
ExceptionSchedule[2]	BACnetARRAY[N] of BACnetSpecialEvent	**String**	Read/Write	No
ObjectIdentifier[3] .ObjectInstance .ObjectType	BACnetObjectIdentier	**DWord**, Long **DWord**, Long **DWord**, Long	Read Only[1]	No
ObjectName	CharacterString	**String**	Read Only	No
ObjectType	**BACnetObjectType**[4]	**DWord**, Long	Read Only[1]	No
PresentValue	Any	**Float**	Read/Write	No
PriorityForWriting	Unsigned Integer	**DWord**, Long	Read/Write	No
ProfileName	CharacterString	**String**	Read Only	No
ScheduleDefault	Real	Float	Read Only	No
StatusFlags[5] .InAlarm .Fault .Overridden .OutofService	BACnetStatusFlags	**Word**, Short Boolean Boolean Boolean Boolean	Read Only[1]	Yes
Reliability	**BACnetReliability**	**DWord**, Long	Read Only[1]	No
OutofService	Boolean	**Boolean**	Read/Write	No
WeeklySchedule[6]	BACnetARRAY[7] of BACnetDailySchedule	**String**	Read/Write	No

Notes:

- May not be made writable.
- *For more information, refer to **ExceptionSchedule String Format**.*

- Object Identifier may be viewed as a packed DWord value (high 10 bits are the object type and low 22 bits are the object instance) or as individual tags for object type and instance using optional sub-property fields.

- Properties with enumerated BACnet Data Types are expressed as integer values. *For standard interpretations, refer to **Enumerated Data Types**.*

- Bit string types may be viewed as a packed Word value, whose actual number of meaningful bits depends on the specific property. They may also be viewed as individual bits using optional sub-property fields.

- *For more information, refer to **WeeklySchedule String Format**.*

Priority Array Elements

Priority Array elements may be "NULL" or the numerical command value currently in effect. The array element index may range from 1 to 16, inclusive.

Trend Log

The following table describes the object's supported properties. The Access column specifies the default access permission for tags. To accommodate non-standard implementations of BACnet, tags may be given Read/Write access unless noted otherwise.

Notes:

1. The length of the array property is specified by *[m]*, where m is the number of supported elements (according to the BACnet specification). The BACnet array properties that do not have a length specified by the BACnet standard is designated by *[N]*. This means the length of the property array depends on the BACnet device. *For more information, refer to **Addressing Examples**.*

2. Implicit Change of Value (COV) notifications are not supported for this object. *For more information on devices with explicit COV support, refer to **COV Notifications**.*

*For more information, refer to **Address Descriptions**, **Addressing Examples**, **COV Notifications**.*

Property Mnemonic	BACnet Data Type	OPC Data Type	Access
AckedTransitions[1] .ToFault .ToNormal .ToOffNormal	BACnetEventTransitionBits	**Word**, Short Boolean Boolean Boolean	Read Only
BufferSize	Unsigned Integer	**DWord**, Long	Read Only
COVResubscriptionInterval	Unsigned Integer	**DWord**, Long	Read/Write
CurrentNotifyTime[2]	BACnetDateTime	**String**	Read Only[3]
Description	CharacterString	**String**	Read Only
EventEnable[1] .ToFault .ToNormal .ToOffNormal	BACnetEventTransitionBits	**Word**, Short Boolean Boolean Boolean	Read/Write
EventState	**BACnetEventState**[4]	**DWord**, Long	Read Only
EventTimeStamps[3]	Array of TimeStamp	**String**	Read Only[3]

Property Mnemonic	BACnet Data Type	OPC Data Type	Access
LastNotifyRecord[5]	Unsigned Integer	**DWord**, Long	Read Only
LogEnable	Boolean	**Boolean**	Read/Write
LogInterval	Unsigned Integer	**DWord**, Long	Read/Write
NotificationClass	Unsigned Integer	**DWord**, Long	Read Only
NotificationThreshold	Unsigned Integer	**DWord**, Long	Read Only
NotifyType	**BACnetNotifyType**[4]	**DWord**, Long	Read Only
ObjectIdentifier[6] .ObjectInstance .ObjectType	BACnetObjectIdentifier	**DWord**, Long **DWord**, Long **DWord**, Long	Read Only[3]
ObjectName	CharacterString	**String**	Read Only
ObjectType	**BACnetObjectType**[4]	**DWord**, Long	Read Only[3]
PreviousNotifyTime[2]	BACnetDateTime	**String**	Read Only[3]
ProfileName	CharacterString	**String**	Read Only
RecordCount	Unsigned Integer	**DWord**, Long	Read/Write
RecordsSinceNotification	Unsigned Integer	**DWord**, Long	Read Only
StartTime	BACnetDateTime	**String**	Read Only[3]
StopTime	BACnetDateTime	**String**	Read Only[3]
StopWhenFull	Boolean	**Boolean**	Read/Write
TotalRecordCount	Unsigned Integer	**DWord**, Long	Read Only

Notes:

1. Bit string types may be viewed as a packed Word value, whose actual number of meaningful bits depends on the specific property. They may also be viewed as individual bits using optional sub-property fields.

2. This has been removed from the BACnet 2004 Specification. Support is available for legacy devices only.

3. May not be made writable.

4. Properties with enumerated BACnet Data Types are expressed as integer values. For standard interpretations, refer to **Enumerated Data Types**.

5. The LastNotifyRecord property replaced the PreviousNotifyTime and CurrentNotifyTime properties in the BACnet 2004 Specification. Devices that support LastNotifyRecord may not support the CurrentNotifyTime and LastNotifyTime properties.

6. Object Identifier may be viewed as a packed DWord value (high 10 bits are the object type and low 22 bits are the object instance) or as individual tags for object type and instance using optional sub-property fields.

Priority Array Elements

Priority Array elements may be "NULL" or the numerical command value currently in effect. The array element index may range from 1 to 16, inclusive.

DateList String Format

The format of the Calendar Object's DateList property string is as follows:

- Entry;Entry;Entry;...Entry;

  - **Date Entry Format**: *0,dDDMMYYYY*, where:

    - *d* is the day of the week, where:

      - 1 to 7 is Monday to Sunday
      - * is unspecified

    - *DD* is the day of the month, where:

      - 01 to 31 is the day of the month
      - 32 is the last day of the month
      - 33 is odd days of the month
      - 34 is even days of the month
      - * * is any day of the month

    - *MM* is the month, where:

      - 01 to 12 is January to December
      - 13 is odd months
      - 14 is even months
      - * * is any month

    - *YYYY* is the year, where:

      - 1900 to 2255 is the year
      - * * * * is any year

  - **Date Range Entry Format**: *1,dDDMMYYYY,dDDMMYYYY*, where:

    - *d* is the day of the week, where:

      - 1 to 7 is Monday to Sunday
      - * is unspecified

    - *DD* is the day of the month, where:

      - 01 to 31 is the day of the month
      - 32 is the last day of the month
      - 33 is odd days of the month
      - 34 is even days of the month
      - * * is any day of the month

    - *MM* is the month, where:

      - 01 to 12 is January to December
      - 13 is odd months

- 14 is even months
- * * is any month
- *YYYY* is the year, where:

  - 1900 to 2255 is the year
  - * * * * is any year
- **Week And Day Entry Format**: *2,MMWd*, where:

  - *MM* is the month, where:

    - 01 to 12 is January to December
    - 13 is odd months
    - 14 is even months
    - * * is any month
  - *W* is the week, where:

    - 1 to 5 is the first week (days 1 to 7) to the fifth week (days 29 to 31) of the month
    - 6 is the last 7 days of the month
    - * is any week in the month
  - *d* is the day of the week, where:

    - 1 to 7 is Monday to Sunday
    - * is any day of the week

Example

Given the following three entries:

- **Date**: Wednesday, January 1, 2014
- **Date Range**: Monday, February 17, 2014 to Friday, February 21, 2014
- **Week And Day**: Any Month, Last 7 days, Friday

The DateList string would be "0,301012014;1,117022014,521022014;2,* * 65;".

ExceptionSchedule String Format

The format of the Calendar Object's ExceptionSchedule property string is as follows:

- Entry;Entry;Entry;...Entry;

  - **Entry Format**: *Period,Priority,Time,Datatype,Length,Value,Time,Datatype,Length,Value,...* where:

    - *Period* is:

      - A Date Entry, Date List Entry, or Week and Day Entry per the **DateList String Format**
      - A Calendar Reference Entry with the format *3,CalRef*, where:

        - *CalRef* is the calendar reference's Object ID

- *Priority* is the BACnet priority (1 to 16)
- *Time* is in the format *HHmmsshh*, where:

  - *HH* is the hour (0 to 23)
  - *mm* is the minute (0 to 59)
  - *ss* is the seconds (0 to 59)
  - *hh* is hundredths of a second (0 to 99)

    Note: All fields may be replaced by * * to mean "any".

- *Datatype* is any BACnet primitive data type, including:

  - 0 is NULL
  - 1 is Boolean
  - 2 is Unsigned Integer
  - 3 is Signed Integer
  - 4 is Real
  - 5 is Double
  - 6 is Octet String
  - 7 is Character String
  - 8 is Bit String
  - 9 is Enumeration
  - 10 is Date. *For more information, refer to* **DateList String Format***.*
  - 11 is Time. *For more information, refer to Time above.*
  - 12 is Object Identifier, where the format is:

    - *<ObjectType>.<Instance>* where *<ObjectType>* is one of the following:

      - AnalogInput
      - AnalogOutput
      - AnalogValue
      - BinaryInput
      - BinaryOutput
      - BinaryValue
      - Calendar
      - Command
      - Device
      - EventEnrollment
      - File
      - Group
      - Loop
      - MultistateInput
      - MultistateOutput
      - NotificationClass

- Program
- Schedule
- Averaging
- MultistateValue
- TrendLog
- LifeSafetyPoint
- LifeSafetyZone
- Accumulator
- PulseConverter
- EventLog
- GlobalGroup
- TrendLogMultiple
- LoadControl
- StructuredView
- AccessDoor
- AccessCredential
- AccessPoint
- AccessRights
- AccessUser
- AccessZone
- CredentialDataInput
- NetworkSecurity
- BitStringValue
- CharacterStringValue
- DatePatternValue
- DateValue
- DateTimePatternValue
- DateTimeValue
- IntegerValue
- LargeAnalogValue
- OctetStringValue
- PositiveIntegerValue
- TimePatternValue
- TimeValue
- NotificationForwarder
- AlertEnrollment
- Channel
- LightingOutput

- *Length* is *n*, where *n* is the number of characters in the value
- *Value* is any primitive data type

Example

Given the following four entries:

- **Date**: Wednesday, January 1, 2014, Priority 16, 01:02:03.00 AM Real "1.23"
- **Date Range**: Monday, February 17, 2014 to Friday, February 21, 2014, Priority 16, 04:05:06.00 AM Real "4.56"
- **Week and Day**: Any Month, Last 7 days, Friday, Priority 16, 07:08:09.00 AM Real "6.78"
- **Calendar Reference**: Calendar Object ID 1, Priority 16, 10:11:12.00 AM Real "9.01"

The ExceptionSchedule string would be
"0,301012014,16,01020300,4,8,1.230000;1,117022014,521022014,16,04050600,4,8,4.560000;2,* * 65,16,07-080900,4,8,6.780000;3,1,16,10111200,4,8,9.010000;".

Prescale String Format

Prescale presents the coefficients used to convert the pulse signals generated by the measuring instrument into the value displayed by PresentValue. This conversion factor is expressed as a ratio of integers – counts out : pulses in.

The format of the Accumulator Object's Prescale property string is as follows:

Multiplier,ModuloDivide
where:
Multiplier = The numerator or counts out; and
ModuloDivide = The denominator or pulses in.

Examples:

- For every 1 pulse in, the present value shows 1 count out. This 1:1 conversion factor has both the multiplier and modulodivide of 1. The string format is: 1,1.
- For every 100 pulses in, the present value shows 1 count out. This 1:100 conversion factor has a multiplier of 1 and a modulodivide of 100. The string format is: 1,100.
- For every 1000 pulses in, the present value shows 10 counts out. This 10:1000 conversion factor has a multiplier of 10 and a modulodivide of 1000. The string format is: 10,1000.

Scale String Format

The format of the Accumulator Object's Scale property string is as follows:

Choice,Value
where:
Choice = Either 0 for float scale or 1 for integer scale; and
Value = The real or integer value depending on the choice.

Examples:

- The string format for a float scale of 1.5 is: 0,1.5.
- The string format for a float scale of 10 is: 0,10.
- The string format for an integer scale of 1 is: 1,1.
- The string format for an integer scale of 10: 1,10.

WeeklySchedule String Format

The format of the Schedule Object's WeeklySchedule property string is as follows:

- Entry;Entry;Entry;Entry;Entry;Entry;Entry;

    - **Entry Format**: *Time,Datatype,Length,Value,Time,Datatype,Length,Value,...* , where:

        - *Time* is in the format *HHmmsshh*, where:

            - *HH* is the hour (0 to 23)
            - *mm* is the minute (0 to 59)
            - *ss* is the seconds (0 to 59)
            - *hh* is hundredths of a second (0 to 99)

            Note: All fields may be replaced by * * to mean "any".

        - *Datatype* is any BACnet primitive data type, including:

            - 0 is NULL
            - 1 is Boolean
            - 2 is Unsigned Integer
            - 3 is Signed Integer
            - 4 is Real
            - 5 is Double
            - 6 is Octet String
            - 7 is Character String
            - 8 is Bit String
            - 9 is Enumeration
            - 10 is Date. *For more information, refer to* **DateList String Format**.
            - 11 is Time. *For more information, refer to Time above.*
            - 12 is Object Identifier, where the format is:

                - *<ObjectType>.<Instance>* where *<ObjectType>* is one of the following:

                    - AnalogInput
                    - AnalogOutput
                    - AnalogValue
                    - BinaryInput
                    - BinaryOutput
                    - BinaryValue
                    - Calendar
                    - Command
                    - Device
                    - EventEnrollment
                    - File
                    - Group
                    - Loop
                    - MultistateInput

- MultistateOutput
- NotificationClass
- Program
- Schedule
- Averaging
- MultistateValue
- TrendLog
- LifeSafetyPoint
- LifeSafetyZone
- Accumulator
- PulseConverter
- EventLog
- GlobalGroup
- TrendLogMultiple
- LoadControl
- StructuredView
- AccessDoor
- AccessCredential
- AccessPoint
- AccessRights
- AccessUser
- AccessZone
- CredentialDataInput
- NetworkSecurity
- BitStringValue
- CharacterStringValue
- DatePatternValue
- DateValue
- DateTimePatternValue
- DateTimeValue
- IntegerValue
- LargeAnalogValue
- OctetStringValue
- PositiveIntegerValue
- TimePatternValue
- TimeValue
- NotificationForwarder
- AlertEnrollment
- Channel
- LightingOutput

- *Length* is *n*, where *n* is the number of characters in the value.
- *Value* is any primitive data type.

Example

Given the following weekly schedule:

- **Monday**: 12:35:50.00 AM Null, 11:59:59.99 PM Boolean "0"
- **Tuesday**: 05:06:07.00 AM Unsigned Integer "12345", 12:22:34.00 PM Signed Integer "-9876"
- **Wednesday**: 12:27:33.00 PM Real "1.234568", 12:28:03.00 PM Double "9.876543"
- **Thursday**: Empty.
- **Friday**: 12:28:45.00 PM Octet String "0123456789ABCDEF", 12:29:11.00 PM Character String "Hello World!"
- **Saturday**: 12:30:51.00 PM Bit String "10,0101010101", 12:32:15.00 PM Enumeration "42"
- **Sunday**: 12:31:46.00 PM Date "03/28/2014", 12:32:15.00 PM Time "12:32:15.00 PM", 12:33:22.00 Object Reference "AnalogInput 12"

The weekly schedule string would be "00355000,0,0,,23595999,1,1,0;05060700,2,5,12345,12223400,3,5,-9876;12273300,4,8,1.234568,12280300,5,8,9.876543;;12284500,6,16,0123456789ABCDEF,12291100,7,12,-Hello
World!;12305100,8,10,0101010101,12321500,9,2,42;12314600,10,9,* 28032014,12321500,11,8,12321500,1-2332200,12,14,AnalogInput,12;".

VBA Scripts for String Parsing and Construction

The Visual Basic for Applications (VBA) code below will convert strings that conform to the **DateList**, **ExceptionSchedule**, and **WeeklySchedule** formats into VBA structures. It will also convert the VBA structures back into appropriately-formatted strings. This can serve as a starting point for Human Machine Interfaces (HMIs) that need to access the BACnet string data in the server and support the use of VBA for scripting.

Note: The following are not supported by this sample script:

- Wildcards ('*') in any BACnetDate or BACnetDateRange contained in a DateList.
- Wildcards ('*') in any BACnetDate or BACnetDateRange that is used to specify the period for an ExceptionSchedule.

```
' --------------------------------------------------------------------------

' KepBacNetApi

'

' Code for converting BACnet/IP DateList, WeeklySchedule, and

' ExceptionSchedule tag strings to VBA types, and back.

'

' Copyright (c) PTC / Kepware, Inc

' --------------------------------------------------------------------------



' Types & Constants ---------------------------------------------------------



' Define constants for the date-list entry type indexes

Public Enum DateEntryTypes

        DateType = 0

        DateRangeType = 1

        WeekNDayType = 2

        CalendarType = 3

End Enum



Type BacNetDate

        CalendarDate As Date

        DayOfWeek As String

End Type
```

```
Type BacNetDateRange

        StartDate As BacNetDate

        EndDate As BacNetDate

End Type


Type BacNetWeekNDay

        Month As String

        Week As String

        Day As String

End Type


Type BacNetDateListEntry

        BDateType As DateEntryTypes

        BDate As BacNetDate

        BDateRange As BacNetDateRange

        BWeekNDay As BacNetWeekNDay

        BCalRef As Integer

End Type


Type BacNetTimeValuePair

        BTime As String

        BDataType As Integer

        BData As String

End Type


Type BacNetDailySchedule

        BTimeValuePairs() As BacNetTimeValuePair

End Type
```

```
Type BacNetException

        BPeriod As BacNetDateListEntry

        BPriority As Integer

        BTimeValuePairs() As BacNetTimeValuePair

End Type



' ---------------------------------------------------------------------------

' Main API Functions

' ---------------------------------------------------------------------------



' DATE LIST -----------------------------------------------------------------



' Parses the input string and populates an array of Date-List entries

Public Function DateListFromKepString(ByVal inputString As String, _

        ByRef InputArray() As BacNetDateListEntry)


        Dim index As Integer, arrayIndex As Integer

        Dim entry As BacNetDateListEntry


        ' Initialize indices that track the string and array positions

        index = 1

        arrayIndex = 0


        ' Clear the input array

        Erase InputArray


        ' Iterate over the entire input string

        While index <= Len(inputString)

                ' Initialize the current date-list entry object
```

```
                    entry = DateListEntryFromKepString(inputString, index)

                    index = index + 1


                    ' Add space for the new array element

                    ReDim Preserve InputArray(0 To arrayIndex)


                    ' Add the entry to the array, and increment the index

                    InputArray(arrayIndex) = entry

                    arrayIndex = arrayIndex + 1

          Wend

End Function


' Returns a KEPServerEX DateList string representing the data in a Date-List

Public Function DateListToKepString( _

          ByRef InputArray() As BacNetDateListEntry) As String


          Dim kepString As String, entryKepString As String

          Dim i As Integer, size As Integer


          kepString = ""

          size = 0


          ' Check the size of the array, on error: skip to the next line of code

          On Error Resume Next

                    size = UBound(InputArray) + 1


          ' If the array isn't empty

          ' convert each date-list entry to its KEPServerEX string format

          If size > 0 Then
```

```
                    ' Iterate through each date-list entry

                For i = LBound(InputArray) To (UBound(InputArray))

                        ' Append the entry string and a delimiting semicolon

                        kepString = kepString & _

                                DateListEntryToKepString(InputArray(i)) & ";"

                Next

        End If



        ' Return the overall string

        DateListToKepString = kepString

End Function



' WEEKLY SCHEDULE ------------------------------------------------------------



' Parses the input string and populates the provided Weekly-Schedule

Public Function WeeklyScheduleFromKepString(ByVal inputString As String, _

        ByRef InputArray() As BacNetDailySchedule)



        Dim dailySchedule As BacNetDailySchedule

        Dim index As Integer, arrayIndex As Integer



        ' Initialize indices that track the string and array positions

        index = 1

        arrayIndex = 0



        ' Clear the input array

        Erase InputArray



        ' Iterate through the entire input string

        While index <= Len(inputString)
```

```
                    ' Populate the dailySchedule's array of time-value pairs

                    TimeValuePairsFromKepString inputString, index, _

                            InputArray:=dailySchedule.BTimeValuePairs


                    index = index + 1


                    ' Add a space in the array of daily-schedules for the current entry

                    ReDim Preserve InputArray(0 To arrayIndex)


                    ' Add the entry to the array, and increment the index

                    InputArray(arrayIndex) = dailySchedule

                    arrayIndex = arrayIndex + 1

            Wend
End Function



' Returns a KEPServerEX WeeklySchedule string representing the data in the

' provided Weekly-Schedule

Public Function WeeklyScheduleToKepString( _

        ByRef InputArray() As BacNetDailySchedule) As String


        Dim kepString As String


        kepString = ""


        ' Iterate through all of the daily-schedule entries in the array

        For i = LBound(InputArray) To (UBound(InputArray))

                ' Append the current daily-schedule KEPServerEX string and a

                ' delimiting semicolon to the overall string

                kepString = kepString & _
```

```
                        TimeValuePairsToKepString(InputArray(i).BTimeValuePairs) & ";"


        Next


        ' Return the overall string

        WeeklyScheduleToKepString = kepString

End Function




' EXCEPTION SCHEDULE -------------------------------------------------------



' Parses the input string and populates the provided Exception-Schedule

Public Function ExceptionScheduleFromKepString( _

        ByVal inputString As String, ByRef InputArray() As BacNetException)


        Dim exception As BacNetException

        Dim priorityTemp As String

        Dim index As Integer, arrayIndex As Integer

        Dim comma() As Variant, commaAndSemi() As Variant


        ' PTC Kepware string delimiters

        comma = Array(",")

        commaAndSemi = Array(",", ";")


        ' Initialize indices that track the string and array positions

        index = 1

        arrayIndex = 0


        ' Clear the input array

        Erase InputArray
```

```
        ' Iterate through the entire input string

        While index <= Len(inputString)

                ' Parse the date-list entry at the start of the current exception

                exception.BPeriod = DateListEntryFromKepString(inputString, index)

                index = index + 1


                ' Parse the priority of the current exception

                priorityTemp = ReadUntilAny(inputString, index, commaAndSemi)

                index = index + Len(priorityTemp)

                exception.BPriority = CInt(priorityTemp)


                ' Parse and populate current exception's array of time-value pairs

                TimeValuePairsFromKepString inputString, index, _

                InputArray:=exception.BTimeValuePairs


                index = index + 1


                ' Add a space in the array of exceptions for the current entry

                ReDim Preserve InputArray(0 To arrayIndex)


                ' Add the entry to the array, and increment the index

                InputArray(arrayIndex) = exception

                arrayIndex = arrayIndex + 1

        Wend

End Function


' Returns a KEPServerEX ExceptionSchedule string representing the data in

' the provided Exception-Schedule

Public Function ExceptionScheduleToKepString( _
```

```
        ByRef InputArray() As BacNetException) As String


    Dim kepString As String

    Dim size As Integer

    Dim tvp As String


    kepString = ""

    size = 0


    ' Check the size of the array, on error: skip to the next line of code

    On Error Resume Next

            size = UBound(InputArray) + 1


    ' If the array isn't empty, then convert each time-value pair to its

    ' KEPServerEX string format

    If size > 0 Then

            ' Iterate through all of the exception entries in the array

            For i = LBound(InputArray) To (UBound(InputArray))

                ' Append the current date-list entry and priority

                 kepString = kepString & _

                    DateListEntryToKepString(InputArray(i).BPeriod) & "," & _

                    InputArray(i).BPriority


                ' If the time-value pair string isn't empty,

                ' append it to the overall string

                tvp = TimeValuePairsToKepString(InputArray(i).BTimeValuePairs)

                If Len(tvp) <> 0 Then

                    kepString = kepString & "," & tvp

                End If
```

```
                            ' add the delimiting semicolon

                            kepString = kepString & ";"

                    Next

            End If



            ' Return the overall string

            ExceptionScheduleToKepString = kepString

End Function



' ------------------------------------------------------------------------------

' INTERNAL FUNCTIONS

' ------------------------------------------------------------------------------



' DATE CONVERSION -------------------------------------------------------------



' Parses the input string and returns a BACnet Date type variable

Private Function DateFromKepString(ByVal inputString As String) _

        As BacNetDate



        Dim DayOfWeek As String

        Dim Day As Integer, Month As Integer, year As Integer

        Dim BDate As BacNetDate



        ' PTC Kepware BACnet string format for a date is: dDDMMYYYY

        ' Extract the segments of the date string

        DayOfWeek = Left(inputString, 1)

        Day = CInt(Mid(inputString, 2, 2))

        Month = CInt(Mid(inputString, 4, 2))

        year = CInt(Right(inputString, 4))
```

```vb
        ' Set the object properties

        BDate.CalendarDate = DateSerial(year, Month, Day)

        BDate.DayOfWeek = DayOfWeek


        ' Return the date type

        DateFromKepString = BDate

End Function



' Returns a PTC Kepware style string that represents the data in the

' provided BACnet Date variable

Private Function DateToKepString(BDate As BacNetDate) As String

        DateToKepString = DateEntryTypes.DateType & "," & _

                DateToKepStringNoId(BDate)



End Function



' Returns a PTC Kepware style string that represents the data in the

' provided BACnet Date variable.

' The Date type ID is not included in the returned string

Private Function DateToKepStringNoId(BDate As BacNetDate) As String

        DateToKepStringNoId = BDate.DayOfWeek & _

                Format(BDate.CalendarDate, "ddmmyyyy")



End Function



' DATE RANGE CONVERSION ------------------------------------------------



' Parses the input string and returns a BACnet Date-Range type variable

Private Function DateRangeFromKepString(ByVal inputString As String) As _
```

```
        BacNetDateRange


        Dim BDateRange As BacNetDateRange


        ' PTC Kepware string format for a date range is: dDDMMYYYY,dDDMMYYYY

        ' Set the start and end date properties

        BDateRange.StartDate = DateFromKepString(Left(inputString, 9))

        BDateRange.EndDate = DateFromKepString(Right(inputString, 9))


        ' Return the date-range type

        DateRangeFromKepString = BDateRange
End Function


' Returns a PTC Kepware style string that represents the data in the

' provided BACnet Date-Range variable

Private Function DateRangeToKepString(BDateRange As BacNetDateRange) _

        As String


        DateRangeToKepString = DateEntryTypes.DateRangeType & "," & _

                DateToKepStringNoId(BDateRange.StartDate) & "," & _

                DateToKepStringNoId(BDateRange.EndDate)


End Function


' WEEK-AND-DAY CONVERSION -------------------------------------------------


' Parses the input string and returns a BacNetWeekNDay type

Private Function WeekNDayFromKepString(ByVal inputString As String) _

        As BacNetWeekNDay
```

```
        Dim BWeekNDay As BacNetWeekNDay


        ' PTC Kepware string format for a week-and-day is: MMWd

        ' Set the object properties by extracting data from the string

        BWeekNDay.Month = Left(inputString, 2)

        BWeekNDay.Week = Mid(inputString, 3, 1)

        BWeekNDay.Day = Right(inputString, 1)


        ' Return the week-and-day type

        WeekNDayFromKepString = BWeekNDay

End Function


' Returns a PTC Kepware style string that represents the data in the

' provided BACnet Week-and-Day variable

Private Function WeekNDayToKepString(BWeekNDay As BacNetWeekNDay) As String

        WeekNDayToKepString = DateEntryTypes.WeekNDayType & "," & _

                BWeekNDay.Month & BWeekNDay.Week & BWeekNDay.Day


End Function


' CALENDAR REFERENCE CONVERSION --------------------------------------------


' Parses the input string and returns a BACnet Calendar object ID

Private Function CalendarReferenceFromKepString( _

        ByVal inputString As String) As Integer


        CalendarReferenceFromKepString = CInt(inputString)

End Function
```

```
' Returns a KEPServerEX style string that represents a BACnet Calendar

' Reference, using the supplied integer value as the calendar ID

Private Function CalendarReferenceToKepString(BCalRef As Integer) As String

        CalendarReferenceToKepString = DateEntryTypes.CalendarType & "," & _

                Format(BCalRef, "0")

End Function



' DATE LIST ENTRY CONVERSION ------------------------------------------------



' Parses the input string starting at the specified index and returns a

' BacNetDateListEntry

Private Function DateListEntryFromKepString(ByVal inputString As String, _

        ByRef stringIndex As Integer) As BacNetDateListEntry



        Dim date1 As String, date2 As String

        Dim comma() As Variant, commaAndSemi() As Variant

        Dim entry As BacNetDateListEntry



        ' PTC Kepware string delimiters

        comma = Array(",")

        commaAndSemi = Array(",", ";")



        ' Get the date-list entry type

        entry.BDateType = CInt(Mid(inputString, stringIndex, 1))

        stringIndex = stringIndex + 2



        ' Based on the entry type from above, populate the appropriate variable

        ' inside the BacNetDateListEntry

        Select Case entry.BDateType
```

```
                ' EntryType is a Date

            Case DateEntryTypes.DateType

                    date1 = ReadUntilAny(inputString, stringIndex, commaAndSemi

                    stringIndex = stringIndex + Len(date1)

                    entry.BDate = DateFromKepString(date1)


            ' EntryType is a Date Range

            Case DateEntryTypes.DateRangeType

                date1 = ReadUntilAny(inputString, stringIndex, comma)

                stringIndex = stringIndex + Len(date1) + 1

                date2 = ReadUntilAny(inputString, stringIndex, commaAndSemi)

                stringIndex = stringIndex + Len(date2)

                entry.BDateRange = DateRangeFromKepString(date1 & "," & date2)


            ' EntryType is a Week-And-Day

            Case DateEntryTypes.WeekNDayType

                date1 = ReadUntilAny(inputString, stringIndex, commaAndSemi)

                stringIndex = stringIndex + Len(date1)

                entry.BWeekNDay = WeekNDayFromKepString(date1)


            ' EntryType is a Calendar Reference

            Case DateEntryTypes.CalendarType

                date1 = ReadUntilAny(inputString, stringIndex, commaAndSemi)

                stringIndex = stringIndex + Len(date1)

                entry.BCalRef = CalendarReferenceFromKepString(date1)

        End Select


        ' Return the date-list entry

        DateListEntryFromKepString = entry

End Function
```

```
' Returns a date-list entry in its KEPServerEx string format

Private Function DateListEntryToKepString(entry As BacNetDateListEntry) _

        As String



        Select Case entry.BDateType

                Case DateEntryTypes.DateType

                    DateListEntryToKepString = DateToKepString(entry.BDate)

                Case DateEntryTypes.DateRangeType

                    DateListEntryToKepString = _

                        DateRangeToKepString(entry.BDateRange)



                Case DateEntryTypes.WeekNDayType

                    DateListEntryToKepString = WeekNDayToKepString(entry.BWeekNDay)

                Case DateEntryTypes.CalendarType

                    DateListEntryToKepString = _

                        CalendarReferenceToKepString(entry.BCalRef)



        End Select

End Function



' TIME VALUE PAIR CONVERSION ------------------------------------------------



' Parses the input string and populates the array of time-value pairs

Private Function TimeValuePairsFromKepString(ByVal raw As String, _

        ByRef stringIndex As Integer, ByRef InputArray() As BacNetTimeValuePair)



        Dim tvp As BacNetTimeValuePair

        Dim comma() As Variant, commaAndSemi() As Variant
```

```vba
Dim lengthTemp As String, typeTemp As String

Dim arrayIndex As Integer


' Initialize index that tracks the array position

arrayIndex = 0


' PTC Kepware string delimiters

comma = Array(",")

commaAndSemi = Array(",", ";")


'Reset the input array

Erase InputArray


' Loop through the input string until the end of the string, or the

' current character is a semicolon

While (stringIndex < Len(raw) And Mid(raw, stringIndex, 1) <> ";")

        ' Skip over the time-value-pair separating comma

        If Mid(raw, stringIndex, 1) = "," Then

                stringIndex = stringIndex + 1

        End If


        ' Get the time (VBA doesn't support hundredths of seconds, so the

        ' time is stored as a string)

        tvp.BTime = Mid(raw, stringIndex, 8)

        stringIndex = stringIndex + 9


        ' Get data type

        typeTemp = ReadUntilAny(raw, stringIndex, comma)

        stringIndex = stringIndex + Len(typeTemp) + 1

        tvp.BDataType = CInt(typeTemp)
```

```
                      ' Get data length

                      lengthTemp = ReadUntilAny(raw, stringIndex, comma)

                      stringIndex = stringIndex + Len(lengthTemp) + 1


                      ' Get the data value

                      tvp.BData = Mid(raw, stringIndex, CInt(lengthTemp))

                      stringIndex = stringIndex + Len(tvp.BData)


                      ' Add space for the array element

                      ReDim Preserve InputArray(0 To arrayIndex)


                      ' Add the new array element, and increment the array index

                      InputArray(arrayIndex) = tvp

                      arrayIndex = arrayIndex + 1

             Wend

End Function


' Return a time-value pair in its KEPServerEX string format

Private Function TimeValuePairToKepString(tvp As BacNetTimeValuePair) _

        As String


      TimeValuePairToKepString = tvp.BTime & "," & tvp.BDataType & "," & _

              Len(tvp.BData) & "," & tvp.BData

End Function


' Return a array of time-value pairs in its KEPServerEX string format

Private Function TimeValuePairsToKepString( _

        ByRef InputArray() As BacNetTimeValuePair) As String
```

```
Dim kepString As String
Dim i As Integer, size As Integer


kepString = ""
size = 0


' Check the size of the array, if there is an error, skip to the next
' line of code
On Error Resume Next
        size = UBound(InputArray) + 1


' If the array isn't empty, then convert each time-value pair to its
' KEPServerEX string format
If size > 0 Then


        ' Loop through all of the time-value pairs
        For i = LBound(InputArray) To (UBound(InputArray))


            ' Append the current time-value pair string to the overall '
            ' string
             kepString = kepString & TimeValuePairToKepString(InputArray(i))


            ' Add a delimiting comma between time-value pairs
             If i < UBound(InputArray) Then
                    kepString = kepString & ","
                End If
        Next
End If
```

```
        ' Return the overall string

        TimeValuePairsToKepString = kepString

End Function



' Build and return a sub-string by reading the input string until the first

' valid delimiter is found

Private Function ReadUntilAny(ByVal raw As String, ByVal start As Integer, _

        ByRef delimiters() As Variant)


        Dim closest, address As Integer

        Dim delimiter As Variant


        ' Initialize the closest delimiter location to past the last character

        ' in the string

        closest = Len(raw) + 1


        If start > Len(raw) Or start <= 0 Then

                ' Return an empty string if the start address is less than 0, or

                ' past the end of the string

                ReadUntilAny = ""

        Else

                ' Iterate through the specified delimiters

                For Each delimiter In delimiters

                  ' Get the address of the closest instance for the current

                  ' delimiter

                   address = InStr(start, raw, delimiter)


                  ' if the address of the current delimiter is closer than

                  ' previously examined delimiters, save the address

                  If address < closest And address > 0 Then
```

```
                        closest = address - 1

                End If

        Next


        ' return the section of the string from the start address to the

        ' closest valid delimiter

        ReadUntilAny = Mid(raw, start, closest - start + 1)

    End If

End Function
```

VBA Scripts Usage Example

The example below demonstrates how the VBA scripts can be used.

```
    Public Function DateListExample()

    Dim DateList() As BacNetDateListEntry

    Dim data As String

    Dim kepString As String


    ' Sample date-list string

    data = "2,0135;"


    ' Populate the date-list (an array of date-list entries)

    DateListFromKepString data, InputArray:=DateList


    ' Convert the date-list back to a KEPServerEx string

    kepString = DateListToKepString(InputArray:=DateList)

End Function


Public Function WeeklyScheduleExample()

    Dim Weekly() As BacNetDailySchedule
```

```
        Dim data As String

        Dim kepString As String

n

        ' Sample weekly-schedule string

        data = "12345678,2,3,123;;;;;;;;"


        ' Populate the weekly-schedule (an array of daily schedules)

        WeeklyScheduleFromKepString data, InputArray:=Weekly


        ' Convert the weekly-schedule back to a KEPServerEx string

        kepString = WeeklyScheduleToKepString(InputArray:=Weekly)

End Function


Public Function ExceptionScheduleExample()

        Dim ExceptionSchedule() As BacNetException

        Dim data As String

        Dim kepString As String


        ' Sample exception-schedule string

        data = "0,301012014,16,23595999,2,3,123;"


        ' Populate the exception-schedule (an array of exceptions)

        ExceptionScheduleFromKepString data, InputArray:=ExceptionSchedule


        ' Convert the exception-schedule back to a KEPServerEX string

        kepString = ExceptionScheduleToKepString( _

                InputArray:=ExceptionSchedule)

End Function
```

See Also: *VBA Scripts for String Parsing and Construction*

Error Descriptions

Click on the link for a list of cancellation and reject reasons, error classes and codes, or messages.

Cancellation Reasons

Reject Reasons

Error Classes and Codes

Event Log Messages

Cancellation Reasons

The following are standard cancellation reason codes as defined in the BACnet specification.

Code	Description
0	Other
1	Buffer overflow
2	Invalid APDU in this state
3	Preempted by higher priority task
4	Segmentation not supported

Reject Reasons

The following are standard reject reason codes defined in the BACnet specification.

Code	Description
0	Other
1	Buffer overflow
2	Inconsistent parameters
3	Invalid parameter data type
4	Invalid tag
5	Missing required parameter
6	Parameter out of range
7	Too many arguments
8	Unidentified enumeration
9	Unrecognized service

Error Classes and Codes

The following are standard error classes and codes defined in the BACnet specification.

BACnet Error Classes

Class	Description
0	Device
1	Object
2	Property

Class	Description
3	Resources
4	Security
5	Services
6	Virtual Terminal
7	Communication

BACnet Error Codes

Code	Description
0	Other
1	Authentication Failed
2	Configuration In Progress
3	Device Busy
4	Dynamic Creation Not Supported
5	File Access Denied
6	Incompatible Security Levels
7	Inconsistent Parameters
8	Inconsistent Selection Criteria
9	Invalid Data Type
10	Invalid File Access Method
11	Invalid File Start Position
12	Invalid Operator Name
13	Invalid Parameter Data Type
14	Invalid Time Stamp
15	Key Generation Error
16	Missing Required Parameter
17	No Objects Of Specified Type
18	No Space For Object
19	No Space To Add List Element
20	No Space To Write Property
21	No VT Sessions Available
22	Property Is Not A List
23	Object Deletion Not Permitted
24	Object Identifier Already Exists
25	Operational Problem
26	Password Failure
27	Read Access Denied
28	Security Not Supported
29	Service Request Denied
30	Timeout

Code	Description
31	Unknown Object
32	Unknown Property
33	--Enumeration Removed--
34	Unknown VT Class
35	Unknown VT Session
36	Unsupported Object Type
37	Value Out Of Range
38	VT Session Already Closed
39	VT Session Termination Failure
40	Write Access Denied
41	Character Set Not Supported
42	Invalid Array Index
43	COV Subscription Failed
44	Not COV Property
45	Optional Functionality Not Supported
46	Invalid Configuration Data
47	Data Type Not Supported
48	Duplicate Name
49	Duplicate Object ID
50	Property is Not an Array
51	Cancel Buffer Overflow
52	Cancel Invalid APDU in this State
53	Cancel Preempted by Higher Priority Task
54	Cancel Segmentation Not Supported
55	Cancel Proprietary
56	Cancel Other
57	Invalid Tag
58	Network Down
59	Reject Buffer Overflow
60	Reject Inconsistent Parameters
61	Reject Invalid Parameter Data Type
62	Reject Invalid Tag
63	Reject Missing Required Parameter
64	Reject Parameter Out of Range
65	Reject Too Many Arguments
66	Reject Undefined Enumeration
67	Reject Unrecognized Service
68	Reject Proprietary
69	Reject Other

Code	Description
70	Unknown Device
71	Unknown Route
72	Value Not Initialized
73	Invalid Event State
74	No Alarm Configured
75	Log Buffer Full
76	Logged Value Purged
77	No Property Specified
78	Not Configured For Triggered Logging
79	--Reserved for Future Use--
80	Parameter Out of Range
81	--Reserved for Future Use--
82	Busy
83	Communication Disabled

Event Log Messages

The following information concerns messages posted to the Event Log pane in the main user interface. Consult the server help on filtering and sorting the Event Log detail view. Server help contains many common messages, so should also be searched. Generally, the type of message (informational, warning) and troubleshooting information is provided whenever possible.

Unable to bind to local address. | IP = '<address>', Port = <number>.

Error Type:
Error

Possible Cause:

1. More than one channel has been configured to use the same IP and port.

2. There is another application running on the system that has already acquired the indicated IP and port for exclusive use.

Possible Solution:

1. Select another local IP address for one of the offending channels. The computer may need to be multihomed.

2. Shut down the other application.

 See Also:
Configuring Multiple Channels

Path to file is invalid. | Path = '<directory>'.

Error Type:
Error

Error reading tag on device. | Tag address = '<address>', Class = '<class> (<ID>)', Code = '<code> (<ID>)'.

Error Type:
Warning

Possible Cause:

1. Message was corrupted.

2. Segmentation is not supported by the device.

3. ReadPropertyMultiple service not supported by the device.

4. Incorrect BACnet implementation.

Possible Solution:

1. No corrective actions may be needed if subsequent requests succeed.

2. If the request was sent in multiple segments and the device does not support request message segmentation, reconfigure the driver to not segment requests.

3. If the driver is configured to allow multiple item requests and the device does not support the ReadPropertyMultiple service, reconfigure the driver to use single item requests.

4. If this particular request never succeeds and the above possibilities are eliminated, note the error class and code and perform a diagnostics capture of transaction (if possible). Refer to the server help documentation or contact Technical Support.

Note:
The hardware vendor should be able to supply a PICS document that details the device's supported properties.

See Also:

1. Error Classes and Codes

2. APDU Settings

Error reading object list from device. | Class = '<class> (<ID>)', Code = '<code> (<ID>)'.

Error Type:
Warning

Possible Cause:

1. Message was corrupted.

2. Segmentation is not supported by the device.

3. Incorrect BACnet implementation.

Possible Solution:

1. No corrective actions may be needed if subsequent requests succeed.

2. If the request was sent in multiple segments and the device does not support request message segmentation, reconfigure the driver to not segment requests.

3. If this particular request never succeeds and the above possibilities are eliminated, note the error class and code and perform a diagnostics capture of transaction (if possible). Refer to the server help documentation or contact Technical Support.

Note:
The hardware vendor should be able to supply a PICS document that details the device's supported properties.

See Also:

1. Error Classes and Codes

2. APDU Settings

Error reading property list from device. | Object type = <type>, Instance = <instance>, Class = '<class> (<ID>)', Code = '<code> (<ID>)'.

Error Type:
Warning

Possible Cause:

1. Device does not support the ReadPropertyMultiple service or "All" property used by driver to acquire list of properties implemented in object.

2. Message was corrupted.

3. Segmentation is not supported by the device.

4. Incorrect BACnet implementation.

Possible Solution:

1. The driver generates a default list of tags if it fails to acquire a list of implemented properties.

2. No corrective actions may be needed if subsequent requests succeed.

3. If the request was sent in multiple segments and the device does not support request message segmentation, reconfigure the driver to not segment requests.

4. If this particular request never succeeds and the above possibilities are eliminated, note the error class and code and perform a diagnostics capture of transaction (if possible). Refer to the server help documentation or contact Technical Support.

Note:
The hardware vendor should be able to supply a PICS document that details the device's supported properties.

See Also:

1. Error Classes and Codes

2. APDU Settings

Generating default property tags. | Object type = <type>, Instance = <instance>.

Error Type:
Warning

Request rejected by device. |

Error Type:

Warning

Possible Cause:

Research the given reason code.

Possible Solution:

This may indicate a BACnet implementation problem. If this particular request never succeeds, note the reject reason and perform a diagnostics capture of transaction (if possible). Refer to the server help documentation or contact Technical Support.

◆ See Also:

Reject Reasons

Request cancelled by device. |

Error Type:

Warning

Possible Cause:

Research the given reason code.

Possible Solution:

This may indicate a BACnet implementation problem. If this particular request never succeeds, note the reject reason and perform a diagnostics capture of transaction (if possible). Refer to the server help documentation or contact Technical Support.

◆ See Also:

Cancellation Reasons

COV subscription failed for tag on device. | Tag address = '<address>', Class = '<class> (<ID>)', Code = '<code> (<ID>)'.

Error Type:

Warning

Possible Cause:

Research the given error class and code.

Possible Solution:

The device may not support COV for the selected item or not have the resources to service the request at the time it was issued. Consider polling the property.

◆ See Also:

1. Error Classes and Codes

2. COV Settings

Error writing tag on device. | Tag address = '<address>', Class = '<class> (<ID>)', Code = '<code> (<ID>)'.

Error Type:

Warning

Possible Cause:

1. Message was corrupted.

2. Segmentation is not supported by the device.

3. Device does not support the WritePropertyMultiple service.

4. Incorrect BACnet implementation.

Possible Solution:

1. No corrective actions may be needed if subsequent requests succeed.

2. If the request was sent in multiple segments and the device does not support request message segmentation, reconfigure the driver to not segment requests.

3. If the driver has been configured to allow multiple item requests, and the device does not support the WritePropertyMultiple service, reconfigure the driver to use single item requests.

4. If this particular request never succeeds and the above possibilities are eliminated, note the error class and code and perform a diagnostics capture of transaction (if possible). Refer to the server help documentation or contact Technical Support.

Note:

The hardware vendor should be able to supply a PICS document that details the device's supported properties.

See Also:

1. Error Classes and Codes

2. APDU Settings

No objects of types specified in device properties found.

Error Type:

Warning

Polling COV item on device. | COV item = '<name>'.

Error Type:

Warning

Possible Cause:

The request to subscribe to COV update notifications for the given tag failed. The driver polls the device for this property instead. The device may not support the SubscribeCOV service (for properties with implicit COV

support) or the SubscribeCOVProperty service (for all other properties.) The device may not support the addressed property.

Possible Solution:

Check the device PICS statement for supported properties and services. There is no harm in polling for the data. COV may need to be disabled for the property or entire device to prevent this error.

See Also:

COV Notifications

Failed to initialize BACnet client for device. Possible duplicate device ID.

Error Type:

Warning

Possible Cause:

Each device that is network visible from the driver must have a unique combination of network number and device instance.

Possible Solution:

Verify device configurations and resolve any conflicts.

See Also:

Device Setup

Devices '<name>' and '<name>' on channel '<name>' are currently configured with same Device ID. Each Device ID must be unique to its channel.

Error Type:

Warning

Could not create group with duplicate object name. Using generic group name. | Duplicate object name = '<name>'.

Error Type:

Warning

Connection failed. Did not get I-Am from remote device.

Error Type:

Warning

Possible Cause:

1. The network connection between the device and the host PC is broken.

2. The communications parameters configured for the device and driver do not match.

Possible Solution:

1. Verify the cabling between the PC and the PLC device.

2. Verify that the specified communications parameters match those of the device.

Connection failed. Could not read segmentation supported from remote device.

Error Type:
Warning

Possible Cause:
The IP address in the Discovery Device properties may be incorrect.

Possible Solution:

1. Verify the IP address of the device.

2. Enable Device Discovery.

See Also:
Device Discovery

Connection failed. Could not read max APDU length from remote device.

Error Type:
Warning

Possible Cause:
The IP address entered in the Discovery Device properties may be incorrect.

Possible Solution:

1. Verify the IP address of the device.

2. Enable Device Discovery.

See Also:
Device Discovery

Connection failed. Could not read protocol services supported from remote device.

Error Type:
Warning

Possible Cause:
The IP address entered in the Discovery Device properties may be incorrect.

Possible Solution:

1. Verify the IP address of the device.

2. Enable Device Discovery.

● See Also:
Device Discovery

Connection failed. Could not register as foreign device for discovery of remote device.

Error Type:
Warning

Possible Cause:

1. The network connection between the device and the host PC is broken.

2. The BBMD IP specified on the Foreign Device Channel Properties is not correct.

3. The BBMD and driver are not network visible to each other.

Possible Solution:

1. Verify the cabling between the PC and the PLC device.

2. Verify the IP of the BBMD.

3. Ping the BBMD from the driver's host computer. Verify the host and BBMD have the correct default gateway IP configured and that an IP router has been configured to join the subnets.

● See Also:
Foreign Device

Error reading segmentation supported from remote device. Segmentation will not be supported.

Error Type:
Warning

Possible Cause:

1. The device ID is incorrect.

2. The device does not allow reading segmentation support.

Possible Solution:

1. Ensure that the correct device ID is being used.

2. Set the discovery method to utilize Who-Is/I-Am services.

Using modified group name. Invalid characters replaced. | Group name = '%s', Original name = '%s'.

Error Type:
Warning

File exception encountered during tag import.

Error Type:
Warning

No data for device instance found in import file. | Device instance = %d.

Error Type:
Warning

Possible Cause:
The import file specified in the Tag Import page did not contain data for the device specified in the General device page.

Possible Solution:

1. Check the import file name on the Tag Import page and the Device ID on the General device page to verify that they are correct.

2. Check the import file to make sure that expected data was exported to the file.

See Also:

1. Tag Import

2. General Device

Tag import terminated. Could not parse file record. | File record = <number>.

Error Type:
Warning

Possible Cause:
The data at the specified line number in the import file could not be parsed due to an unexpected syntax or record length.

Possible Solution:

1. Verify that the correct import file was specified and that the file was generated by the specified application.

2. Verify that the import file is not corrupted.

3. Edit or recreate the file if necessary.

See Also:
Tag Import Settings

Imported tag database may be incomplete due to communication error.

Error Type:
Warning

Possible Cause:
The driver did not receive a response from the device at some point during the tag import procedure.

Possible Solution:
Retry import. If problem persists, check network hardware.

Device is not responding.

Error Type:
Warning

Possible Cause:

1. The network connection between the device and the host PC is broken.

2. The communications parameters configure for the device and driver do not match.

3. The response from the device took longer to receive than the amount of time specified in the Request Timeout device setting.

Possible Solution:

1. Verify the cabling between the PC and the PLC device.

2. Verify that the specified communications parameters match those of the device.

3. Increase the Request Timeout so that the entire response can be handled.

Error parsing write data for tag. Data does not match DateList format. Refer to help file for more information. | Tag address = '<address>'.

Error Type:
Warning

Possible Cause:
The data for the attempted write is not formatted properly as a DateList string.

Possible Solution:
Update the DateList string to the correct format.

Note:
In the case of a multiple tag write, none of the tags are written to the device if the write value for a DateList Tag fails to parse.

See Also:
DateList String Format

Error parsing write data for tag. Data does not match WeeklySchedule format. Refer to help file for more information. | Tag address = '<address>'.

Error Type:
Warning

Possible Cause:
The data for the attempted write is not formatted properly as a WeeklySchedule string.

Possible Solution:
Update the WeeklySchedule string to the correct format.

Note:
In the case of a multiple tag write, none of the tags are written to the device if the write value for a WeeklySchedule Tag fails to parse.

See Also:
WeeklySchedule String Format

Error parsing write data for tag. Data does not match ExceptionSchedule format. Refer to help file for more information. | Tag address = '<address>'.

Error Type:
Warning

Possible Cause:
The data for the attempted write is not formatted properly as an ExceptionSchedule string.

Possible Solution:
Update the ExceptionSchedule string to the correct format.

Note:
In the case of a multiple tag write, none of the tags are written to the device if the write value for an ExceptionSchedule Tag fails to parse.

See Also:
ExceptionSchedule String Format

Error parsing write data for tag. Data does not match Scale format. Refer to help file for more information. | Tag address = '<address>'.

Error Type:
Warning

Possible Cause:

The data for the attempted write is not formatted properly as a Scale string.

Possible Solution:

Update the Scale string to the correct format.

Note:

In the case of a multiple tag write, none of the tags are written to the device if the write value for a Scale Tag fails to parse.

See Also:

Scale String Format

Error parsing write data for tag. Data does not match Prescale format, refer to help file for more information. | Tag address = '<address>'.

Error Type:

Warning

Possible Cause:

The data for the attempted write is not formatted properly as a Prescale string.

Possible Solution:

Update the Prescale string to the correct format.

Note:

In the case of a multiple tag write, none of the tags are written to the device if the write value for a Prescale Tag fails to parse.

See Also:

Prescale String Format

CSV import failure for notification object identifier list. Incorrect field count.

Error Type:

Warning

Possible Cause:

Importing the notification object identifier list from a CSV file failed due to an incorrect field count.

Possible Solution:

Locate the record that has an incorrect number of fields and add or remove as necessary.

See Also:

CSV Import / Export

CSV import failure for notification object identifier list. Missing object type.

Error Type:

Warning

Possible Cause:

Importing the notification object identifier list from a CSV file failed due to a missing object type.

Possible Solution:

Locate and correct the record with the missing object type.

See Also:

CSV Import / Export

CSV import failure for notification object identifier list. Missing or incorrect CSV file header.

Error Type:

Warning

Possible Cause:

Importing the notification object identifier list from a CSV file failed due to a missing or incorrect CSV file header.

Possible Solution:

Add or correct the CSV file header.

See Also:

CSV Import / Export

CSV import failure for notification object identifier list. Invalid object type. | Type = '<type>'.

Error Type:

Warning

Possible Cause:

Importing the notification object identifier list from a CSV file failed due to an invalid object type.

Possible Solution:

Locate and correct the record with the invalid object type.

See Also:

CSV Import / Export

CSV import failure for notification object identifier list. Invalid object instance. | Instance = '<instance>'.

Error Type:

Warning

Possible Cause:

Importing the notification object identifier list from a CSV file failed due to an invalid object instance.

Possible Solution:

Locate and correct the record with the invalid object instance.

● **See Also:**

CSV Import / Export

CSV import failure for notification object identifier list. No valid records in file.

Error Type:

Warning

Possible Cause:

Importing the notification object identifier list from a CSV file failed because there were no valid records in the file.

Possible Solution:

Ensure that the CSV file has at least one valid record or add one.

● **See Also:**

CSV Import / Export

Importing tag database from file. | Filename = '<name>'.

Error Type:

Informational

PIC Statement

Overview

Date	3May16
Vendor Name	PTC, Inc. Kepware Technologies
Product Names	KEPServerEX®, ThingWorx Kepware Server, ThingWorx Kepware Edge, OPC Aggregator
Product Model Number	BACnet/IP Driver
Application Software Version	5.20 or higher
Firmware Revision	N/A
BACnet Protocol Revision	135-2012

Product Description

The server platform, paired with the BACnet/IP Driver, has several primary use cases:

- Managing non-BACnet devices within a BACnet system. For example, a number of non-BACnet PLCs controlling existing air handling and lighting equipment can feed information into a BACnet SCADA for monitoring systems, analyzing data, and triggering actions based on events.
- Pushing BACnet data into a non-BACnet SCADA. For example, a very large plant-wide or smart city-wide non-BACnet SCADA controlling and monitoring a myriad of operations can use the BACnet Driver to feed BACnet information into the non-BACnet SCADA.
- Accessing BACnet data through the server's IoT Gateway. For example, device data from building automation systems can be sent across web interfaces that utilize standard web connectivity protocols and take advantage of the associated security to then be utilized and combined in an IoT environment.

BACnet Standardized Device Profile (Annex L)

* The server's functionality is not covered by the standard definitions within the BACnet specification's Annex L Descriptions and Profiles of Standardized BACnet devices. The server's primary function is as a gateway for BACnet data among a variety of non-BACnet networks and protocols.

☐	BACnet Operator Workstation (B-OWS)
☐	BACnet Advanced Operator Workstation (B-AWS)
☐	BACnet Operator Display (B-OD)
☐	BACnet Building Controller (B-BC)*
☐	BACnet Advanced Application Controller (B-AAC)
☐	BACnet Application Specific Controller (B-ASC)
☐	BACnet Smart Sensor (B-SS)
☐	BACnet Smart Actuator (B-SA)

BACnet Interoperability Building Blocks Supported (BIBBs)

BIBB*	BACnet Service	Initiate	Execute
DM-DDB-A	Who-Is	X	
DM-DDB-B	Who-Is		X
DM-DDB-A	I-Am		X
DM-DDB-B	I-Am	X	
DS-RP-A	ReadProperty	X	
DS-RP-B	ReadProperty		X
DS-RPM-A	ReadPropertyMultiple	X	
DS-RPM-B	ReadPropertyMultiple		X
DS-WP-A	WriteProperty	X	
DS-WPM-A	WritePropertyMultiple	X	
DS-COV-A	SubscribeCOV	X	
DS-COVP-A	SubscribeCOVProperty	X	
DS-COV-A	ConfirmedCOVNotification		X
DS-COV-A	UnconfirmedCOVNotification		X
AE-N-A	ConfirmedEventNotification Supported Event Types: • Change-of-state • Change-of-value • Command-failure • Out-of-range • Unsigned-range		X
AE-N-A	UnconfirmedEventNotification Supported Event Types: • Change-of-state • Change-of-value • Command-failure • Out-of-range • Unsigned-range		X

Standard Object Types Supported

* The server supports the objects in the table below, but not in the traditional sense of the Object existing on a device. Although the server does have a BACnet device ID, it is actually a gateway that passes BACnet data through. Therefore, all of the supported Objects are vehicles for passing data for that object on a device to a consuming system. The objects are not dynamically creatable or deletable from the server.

Supported Object	Creatable	Deletable
Accumulator		
Analog Input		

Supported Object	Creatable	Deletable
Analog Output		
Analog Value		
Averaging		
Binary Input		
Binary Output		
Binary Value		
Calendar		
Command		
Device		
Event Enrollment		
File		
Group		
Life Safety Point		
Life Safety Zone		
Loop		
Multi-State Input		
Multi-state Value		
Notification Class		
Program		
Schedule		
Trend Log		

Segmentation Capability

X	Able to receive segmented messages, Window Size 1-127 bytes

Data Link Layer Options

X	BACnet IP, (Annex J)

X	BACnet IP, (Annex J), Foreign Device

	ISO 8802-3, Ethernet (Clause 7)

	ATA 878.1, 2.5 Mb. ARCNET (Clause 8)

	ATA 878.1, EIA-485 ARCNET (Clause 8), baud rate(s): ___

	MS/TP master (Clause 9), baud rate(s): ___

	MS/TP slave (Clause 9), baud rate(s): ___

	Point-To-Point, EIA 232 (Clause 10), baud rate(s): ___

	Point-To-Point, modem, (Clause 10), baud rate(s): ___

	LonTalk, (Clause 11), medium: ___

	BACnet/ZigBee (ANNEX O)
	Other: _____

Device Address Binding

Is static device binding supported? (This is currently necessary for two-way communication with MS/TP slaves and certain other devices.)

	Yes
X	No

Networking Options

	Router, Clause 6 - List all routing configurations, e.g., ARCNET-Ethernet, Ethernet-MS/TP, etc.
	Annex H, BACnet Tunneling Router over IP
	BACnet/IP Broadcast Management Device (BBMD)

		Yes		No
Does the BBMD support registrations by Foreign Devices?		Yes	**N/A**	No
Does the BBMD support network address translation?		Yes	**N/A**	No

Network Security Options

X	Non-Secure Device - is capable of operating without BACnet Network Security
	Secure Device - is capable of using BACnet Network Security (NS-SD BIBB)
	Multiple Application-Specific Keys
	Supports Encryption (NS-ED BIBB)
	Key Server (NS-KS BIBB)

Character Sets Supported

X	ISO 10646 (UTF-8)		**X**	IBM/Microsoft DBCS*		**X**	ISO 8859-1
X	ISO 10646 (UCS-2)			ISO 10646 (UCS-4)			JS X 0208

Communication Gateway Protocol Support

The server's primary function is as a gateway for BACnet data across a variety of non-BACnet networks and protocols.

All trademarks and registered trademarks are the property of their corporate owners.

Index

A

B

C

G

Generate 16

Generating default property tags. | Object type = <type>, Instance = <instance>. 126

Global 26, 28

Group 73

H

Help Contents 6

I

I-Am 26, 31

ID 12

Identification 12

Implicit 20

Import Method 25

Imported tag database may be incomplete due to communication error. 133

Importing tag database from file. | Filename = '<name>'. 137

Initial Updates from Cache 14

Instance Number 23

Intrinsic Reporting 23

Invalid 120

Invalid APDU 120

IP Addresses 34

L

Life Safety Point 74

Life Safety Zone 76

List Addresses 50

Local 26, 28

Long 37

Loop 78