KB2010-04: OPTIMIZING THE MBX OPC DRIVER AGENT

Cyberlogic's OPC Server Suites include several unique features that you can adjust to achieve superior communication performance. This article covers the features and techniques that apply to the MBX OPC server. The MBX product line communicates with Schneider Electric's Modicon controllers and networks, as well as Modbus-compliant products from other vendors.

Applies To:

- MBX OPC Server Suite
- MBX OPC Premier Suite
- MBX OPC Enterprise Suite

Issues:

Several sets of parameters and configuration techniques affect communication performance:

- *Device Type:* Modicon controllers limit the number of registers of each type that the OPC server can include in a single message. Improper settings in the server may cause errors or excessive communication overhead.
- *Maximum Concurrent Requests:* These settings, at the network and node levels, control how network resources are allocated among competing devices.
- *Message Blocking:* This setting lets you control the tradeoff between unneeded message content and excessive message overhead.
- *Span Messages:* You can permit the server to break large arrays into multiple messages, which may allow it to transfer their data more efficiently.
- *Unsolicited Communication:* The normal polled mode can be inefficient for data that changes infrequently. To improve on this, you can eliminate the polling, and instead configure the controllers to send their data whenever it changes.

Procedure:

You may adjust the settings in each of the five areas listed above. However, some of them may not apply to your system. For example, Span Messages applies only if you use arrays, and may be insignificant for small arrays.

The following sections will help you decide if you should make any changes to these settings, and give you an overview of how to make them. You can find detailed information for all of these procedures in the *MBX OPC Driver Agent Help*.

Device Type

Each Modicon controller type can handle only a limited number of coils, inputs or registers in a single transaction. If you request more than the maximum, the communication will fail. Device Type specifies these limits for your controller.

For more information about device types, refer to the section *Editing the MBX Driver Agent* in the *Configuration Editor Reference* of the *MBX OPC Driver Agent Help*.

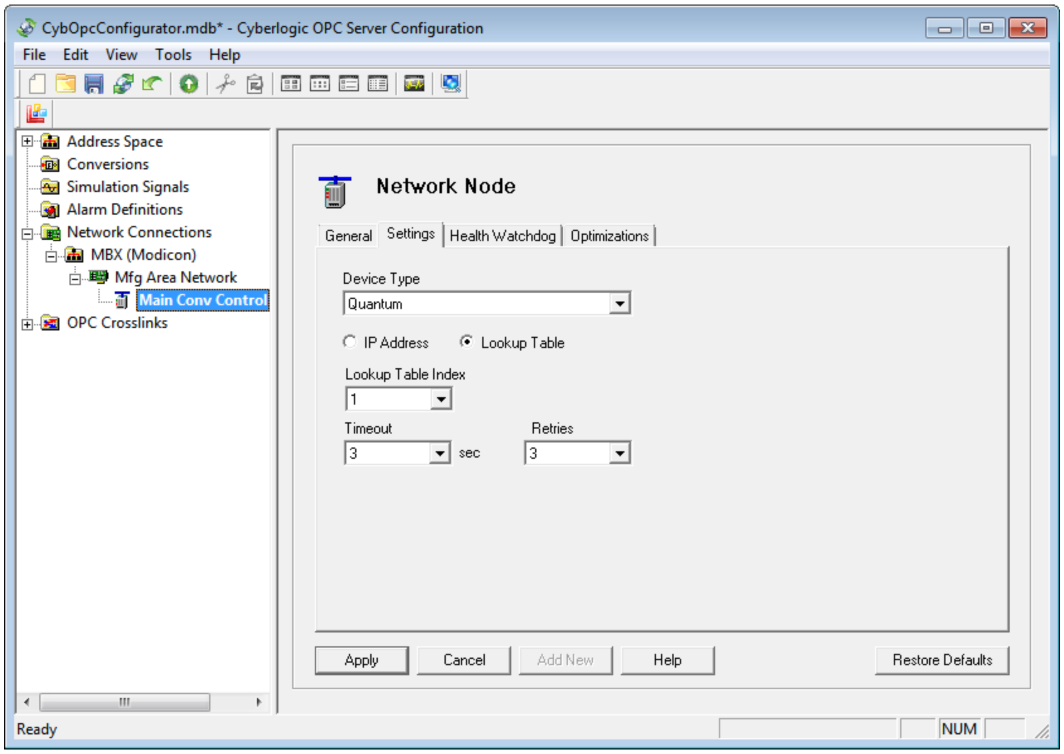
Things to consider

- The default is *Auto Detect*, which tries to detect the controller type, and then picks the proper settings. This is usually the best choice.
- If Auto Detect cannot identify the controller, it selects *Safe Settings*. This will work for all Modicon controllers and most third-party products, but the performance may be very poor.
- Some third-party products do not support Auto Detect. In such a case, you must either select the Modicon controller with the closest settings, or create a custom device.
- The Message Blocking setting may also limit the message size. The server will size the messages to fit within both criteria.

Should I modify the Device Type setting?

Auto Detect uses Modbus function code 17 (*Report Slave ID*) to identify the device type. All Schneider Electric controllers support this function, so Auto Detect is the preferred setting for them. For third-party devices, check with the manufacturer to see if it supports this function. If not, you should create a custom device or choose one of the existing device selections.

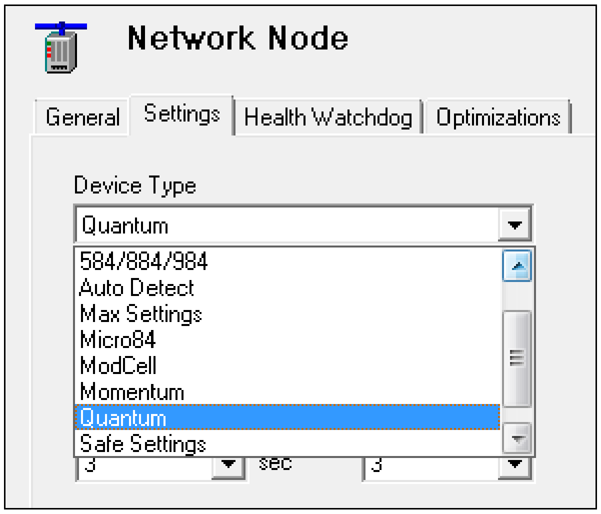
What to do



Device Type: Quantum
IP Address [✓ Lookup Table]
Lookup Table Index: 1
Timeout: 3 sec
Retries: 3

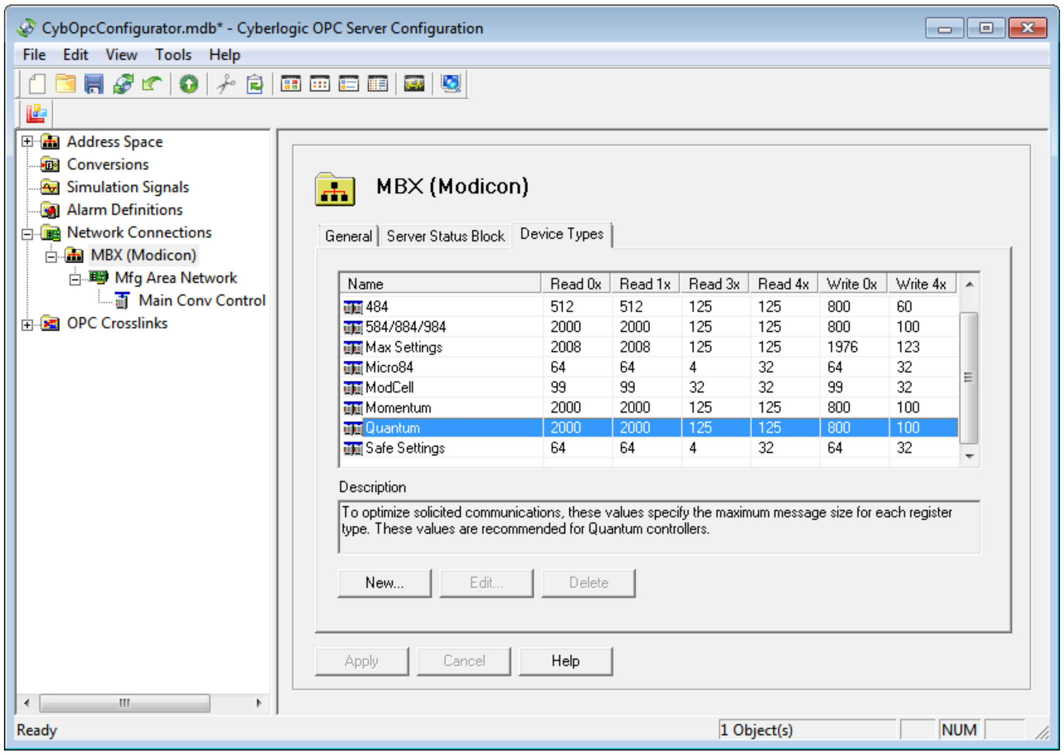
Device Type is a network node setting.

To set a device type, select the ***network node*** and then go to the ***Settings*** tab.



Pick the controller type from the ***Device Type*** drop-down.

Auto Detect is usually the preferred selection for Modicon devices and for third-party devices that support the *Report Slave ID* function, but you may want to specify a particular model instead. Do not use Auto Detect for third-party devices that do not support *Report Slave ID*; it will not detect them.



Name	Read 0x	Read 1x	Read 3x	Read 4x	Write 0x	Write 4x
484	512	512	125	125	800	60
584/884/984	2000	2000	125	125	800	100
Max Settings	2008	2008	125	125	1976	123
Micro84	64	64	4	32	64	32
ModCell	99	99	32	32	99	32
Momentum	2000	2000	125	125	800	100
Quantum	2000	2000	125	125	800	100
Safe Settings	64	64	4	32	64	32

Description

To optimize solicited communications, these values specify the maximum message size for each register type. These values are recommended for Quantum controllers.

If you want to create a new device type, select the ***MBX (Modicon)*** folder under ***Network Connections***. Now go to the ***Device Types*** tab. From here, click the ***New…*** button to begin creating your new device.

Maximum Concurrent Requests

These settings limit the number of simultaneous transactions that the server will request. Maximum Network Requests limits the total number of requests from all nodes on a network, and Maximum Node Requests limits the requests from an individual network node. Together, they allow you to balance the resource use among the nodes on a network.

For a complete discussion with detailed examples, refer to *Appendix E: Configuring Maximum Concurrent Requests* in the *MBX OPC Driver Agent Help*.

Things to consider

- You can set the maximum concurrent requests at both the network connection and network node level.
- Slower networks and slower nodes need lower limits.
- If you set the limit too high, the server may send the node more requests than it can handle, needlessly tying up resources.
- If you set the limit too low, you may prevent the network from achieving its best performance, while leaving available resources unused.

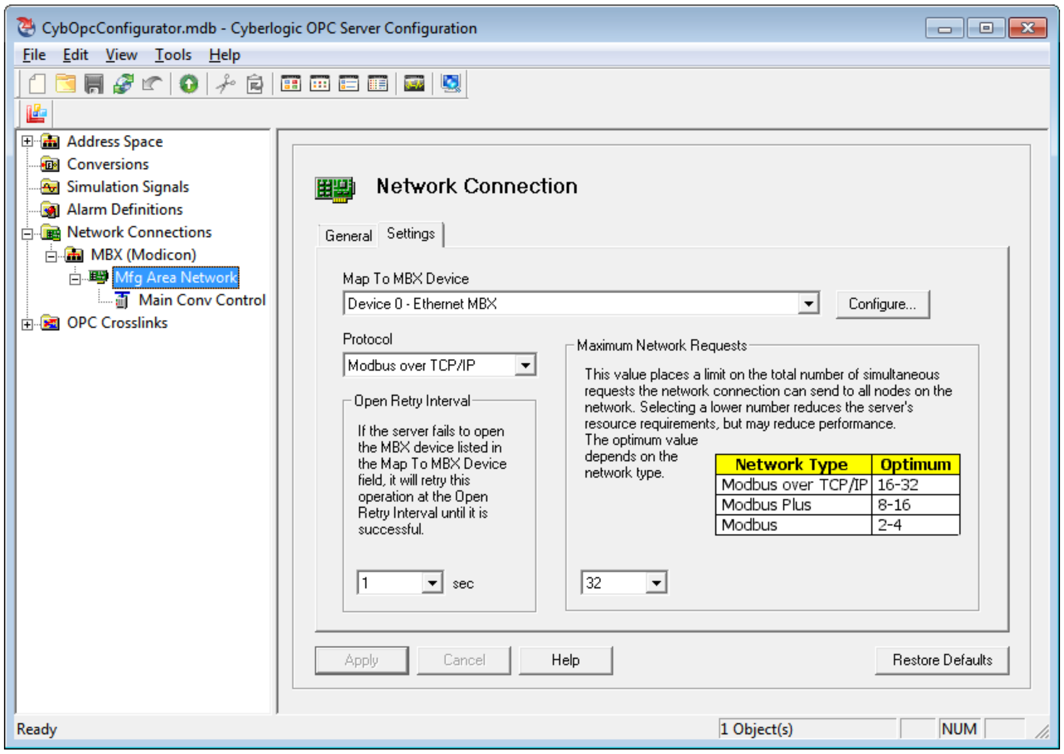
Should I modify the maximum concurrent requests settings?

When you create a network connection or network node, the editor selects an appropriate value that is usually the best choice.

If the messages will pass through a bridge to a slower network, the default value will probably be too high, and you should select a lower setting. Always use the appropriate settings for the slowest network in the chain.

If the network is shared with other applications, you may want to lower the setting to free more resources for the other applications.

What to do



Map To MBX Device: Device 0 - Ethernet MBX

Protocol: Modbus over TCP/IP

Open Retry Interval

If the server fails to open the MBX device listed in the Map To MBX Device field, it will retry this operation at the Open Retry Interval until it is successful.

1 sec

Maximum Network Requests

This value places a limit on the total number of simultaneous requests the network connection can send to all nodes on the network. Selecting a lower number reduces the server's resource requirements, but may reduce performance. The optimum value depends on the network type.

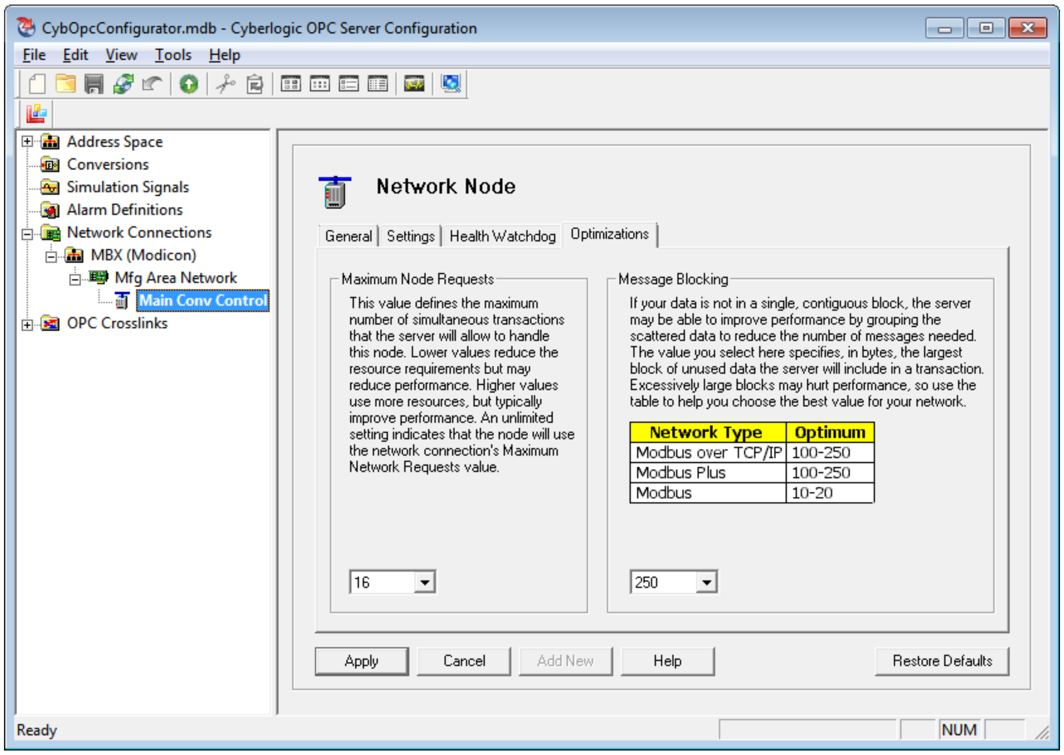
Network Type	Optimum
Modbus over TCP/IP	16-32
Modbus Plus	8-16
Modbus	2-4

32

Maximum Network Requests is a network connection setting.

Maximum Network Requests limits the total number of simultaneous requests that the server will send to all nodes on the network. To change this value, select a ***network connection***, and go to its ***Settings*** tab. The ***Maximum Network Requests*** group provides a drop-down box, which allows you to select the desired setting.

The chart provides a recommended setting range for each network type.



Maximum Node Requests

This value defines the maximum number of simultaneous transactions that the server will allow to handle this node. Lower values reduce the resource requirements but may reduce performance. Higher values use more resources, but typically improve performance. An unlimited setting indicates that the node will use the network connection's Maximum Network Requests value.

16

Message Blocking

If your data is not in a single, contiguous block, the server may be able to improve performance by grouping the scattered data to reduce the number of messages needed. The value you select here specifies, in bytes, the largest block of unused data the server will include in a transaction. Excessively large blocks may hurt performance, so use the table to help you choose the best value for your network.

Network Type	Optimum
Modbus over TCP/IP	100-250
Modbus Plus	100-250
Modbus	10-20

250

Maximum Node Requests is a network node setting.

Maximum Node Requests limits the number of simultaneous requests that the server will send to an individual node. To change this value, select a ***network node***, and go to its ***Optimizations*** tab. The ***Maximum Node Requests*** group provides a drop-down box, which allows you to select the desired setting.

Message Blocking

The server transfers data in blocks of contiguous registers, inputs or coils. However, the data you need may not be contiguous in the controller’s memory. Typically, this means that the server will request some values that are not needed, because doing so can be more efficient than if the server requested numerous smaller blocks of data. The Message Blocking setting lets you limit the amount of unneeded data that may be included in each block.

For a complete discussion of message blocking, refer to the discussion of the *Optimizations Tab* in the *Editing Network Nodes* section of the *MBX OPC Driver Agent Help*.

Things to consider

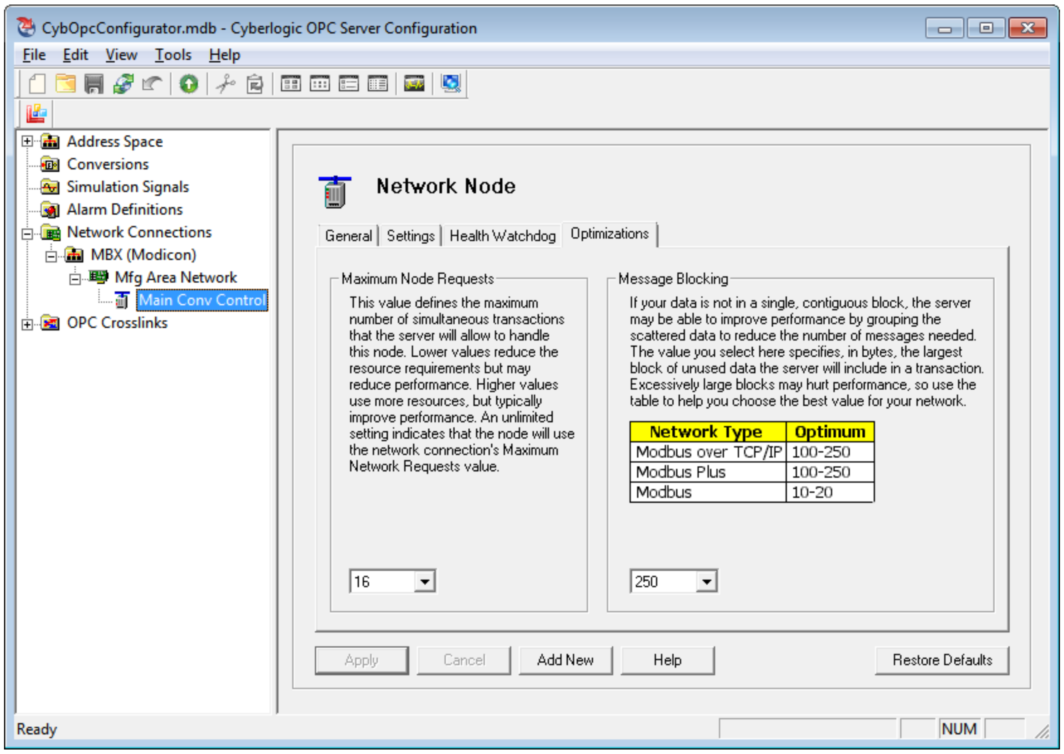
- The tradeoff is between the extra time required to transmit the unneeded data, and the extra time required for overhead in multiple messages.
- Fast networks work best with larger block sizes. The transmission time wasted on the unneeded data is less than the overhead penalty for creating many small messages.
- Slow networks work best with smaller block sizes. The extra time to slowly transmit a large quantity of unneeded data quickly grows to be larger than the overhead for creating a greater number of messages.
- Look at rearranging the memory organization in the controller to minimize gaps in the data you need.
- The Device Type setting also limits the message size. The server will size the messages to fit within both criteria.

Should I modify the message blocking setting?

The editor chooses a setting for the node based on the type of network connection it uses. If the messages must pass through a bridge to a different type of network, the default setting may not be optimal. Select a message blocking value that is in the range for the slower network.

You should also look at how the data you need is distributed in the controller’s memory. That may give you some clues about how to set the blocking. For example, the data you need may be in more or less compact groups that are widely separated from each other. In that case, you would set the blocking to be able to pick up these groups without taking the data in the large gaps between them.

What to do



Maximum Node Requests

This value defines the maximum number of simultaneous transactions that the server will allow to handle this node. Lower values reduce the resource requirements but may reduce performance. Higher values use more resources, but typically improve performance. An unlimited setting indicates that the node will use the network connection's Maximum Network Requests value.

16

Message Blocking

If your data is not in a single, contiguous block, the server may be able to improve performance by grouping the scattered data to reduce the number of messages needed. The value you select here specifies, in bytes, the largest block of unused data the server will include in a transaction. Excessively large blocks may hurt performance, so use the table to help you choose the best value for your network.

Network Type	Optimum
Modbus over TCP/IP	100-250
Modbus Plus	100-250
Modbus	10-20

250

Message Blocking is a network node setting.

To set the Message Blocking, select a ***network node***, and go to its ***Optimizations*** tab. A drop-down box allows you to select the setting you want to use. The value you select is the largest block of unneeded data that will be allowed in a message.

The chart provides a recommended range for each network type.

Span Messages

An array is a group of inputs, coils or registers that the server treats as a unit. When the server sets up a message to read or write the values in an array, it may pass the entire array within a single message, a process known as atomic transfer. Alternatively, it may split the array elements across two or more messages, which is called spanning the messages.

The span messages property lets you give the server permission to split the array up, or force it to use atomic transfers.

Things to consider

- To obtain the best performance, enable spanning whenever possible.
- Disable span messages for an array only if it is critical for all elements in that array to update at the same time. Generally, this will result in lower performance.
- If an array is larger than a single message can carry, you must enable span messages for that array. If you do not, you will get a runtime error.
- Span messages is set individually for each array, so you can allow some to span messages, and force others to use atomic reads and writes.

Should I modify the span messages setting?

This setting applies only to arrays. If none of your data items are configured as arrays, then you can ignore it.

The server enables spanning for all arrays by default. If you have never changed these defaults, then message spanning is already configured to maximize performance.

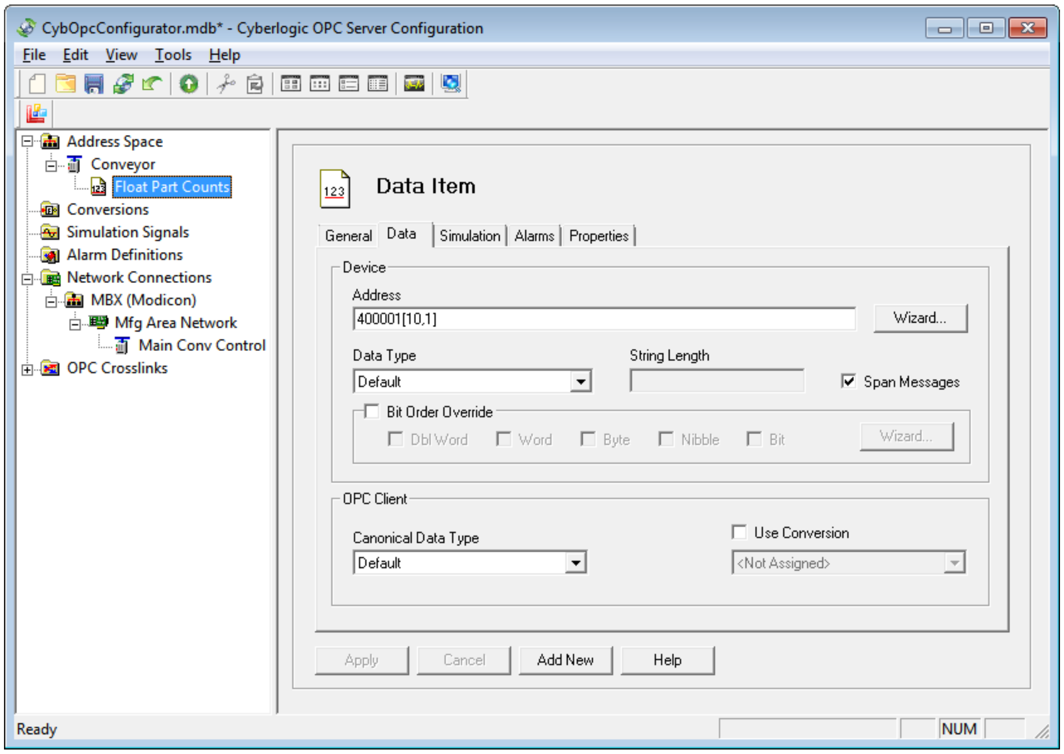
If some of your arrays have span messages disabled, you should determine if atomic transfers are really necessary.

Caution!

You must not disable span messages for arrays that are larger than the maximum size allowed for a single message. Doing so will cause communication errors. Refer to the Device Type settings for your controller to determine the message size limits for each array type.

For example, suppose you are using a Micro 84 controller and have an array of fifty 4xxx registers. A Micro 84 can transfer only 32 of these registers in a single message. Therefore, the array cannot fit into a single message. You must enable span messages for this array.

What to do



Device
Address: 400001[10,1]
Data Type: Default
String Length:
[✓] Span Messages
[ ] Bit Order Override: [ ] Dbl Word [ ] Word [ ] Byte [ ] Nibble [ ] Bit
OPC Client
Canonical Data Type: Default
[ ] Use Conversion: <Not Assigned>

Span Messages is a data item setting.

To enable message spanning, open an ***array data item*** and go to its ***Data*** tab. Check the box for ***Span Messages***.

To disable message spanning and force atomic reads and writes, uncheck the box.

Unsolicited Communication

OPC servers normally operate in polled mode. This means that the server periodically requests data from the controllers or other data sources, and they respond to these requests. If your configuration has a large quantity of data that changes infrequently, polled mode can be very inefficient, because the server will repeatedly poll for values that have not changed.

Unsolicited mode, on the other hand, allows the controllers to send their data whenever it changes. This can dramatically improve efficiency by eliminating unnecessary polling for unchanged data.

For additional information on unsolicited communication, refer to the discussion of the *Unsolicited Message Filters Tab* in the *Devices* section of the *MBX OPC Driver Agent Help*.

Things to consider

- If you use solicited mode and have performance problems, check the rate at which you are polling for new data. Is the polling rate really necessary, or is it excessively high? Reducing the solicited update rate is frequently the easiest way to improve performance.
- An MBX OPC server can use both solicited and unsolicited mode at the same time to communicate with a network node. That means you can poll for data that changes rapidly, and get the rest in unsolicited updates.

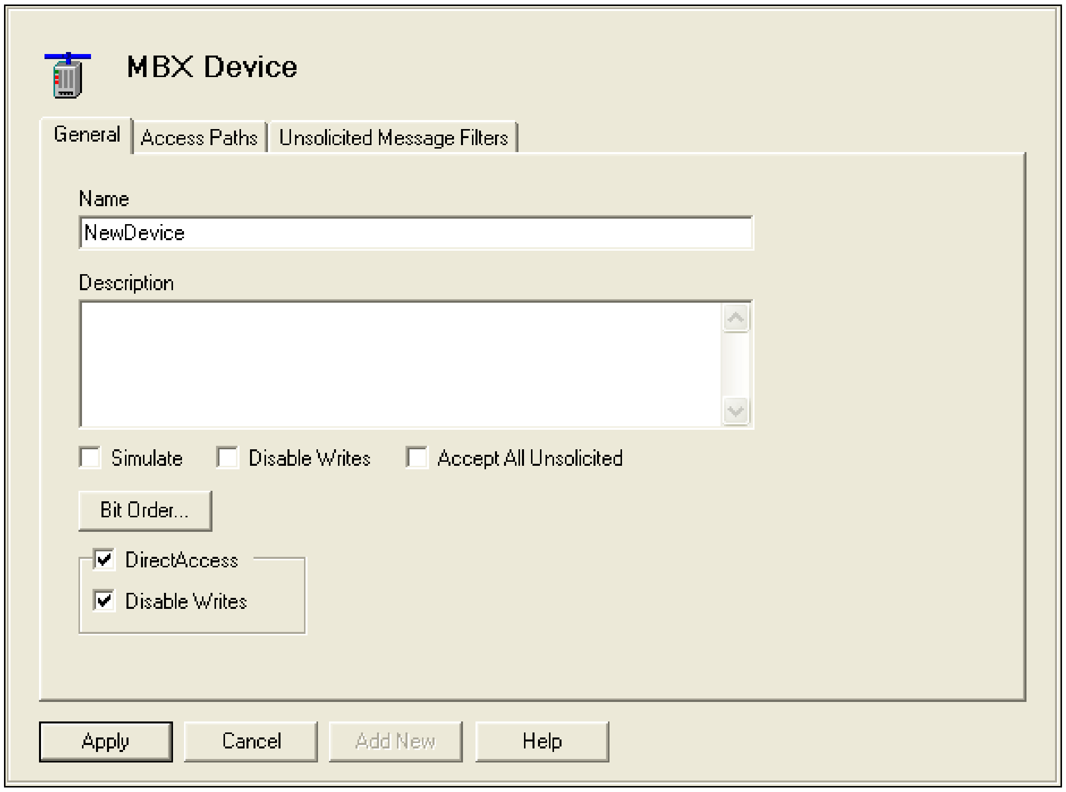
Should I use unsolicited communication?

A configuration with a large quantity of data that changes infrequently will benefit most from unsolicited communication.

If most of your data changes rapidly, you will see little, if any, benefit from a switch to unsolicited mode.

What to do

You must configure the OPC server for unsolicited communication, and then you must edit the ladder logic in the PLC to program the update messages. You can get details on the types of supported messages in *Appendix D: Unsolicited Message Programming* in the *MBX OPC Driver Agent Help*.



Name: NewDevice

Description:

☐ Simulate ☐ Disable Writes ☐ Accept All Unsolicited

☑ DirectAccess
☑ Disable Writes

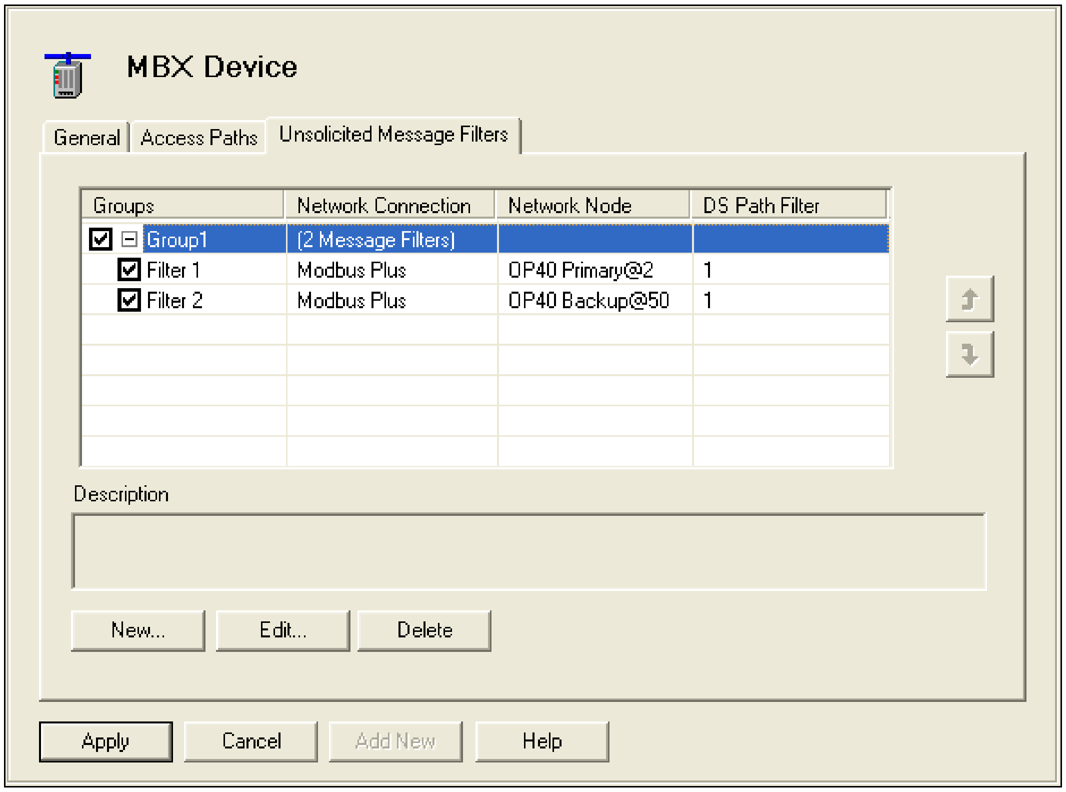
Unsolicited communication is configured in the MBX Device.

You can filter unsolicited messages to ensure that they come only from trusted sources. If you do not want to use the filtering feature, you can disable it. To do this, go to the ***General*** tab for the MBX device and check the ***Accept All Unsolicited*** box.

Note When you check ***Accept All Unsolicited***, the server will receive the unsolicited messages on the slave path(s) specified in the Default DS Paths field on the network connection's General tab.

Most users leave this box unchecked and instead go to the ***Unsolicited Message Filters*** tab and configure filters there.

Note When you configure unsolicited messaging on a device, it is a good idea to check ***Accept All Unsolicited*** during the initial setup, even if you plan to use unsolicited message filters. This allows you to verify that the PLC programming is correct and that the data transfers properly, without being concerned about the filter setup. After you confirm that part of the configuration, you can then clear this box and configure the unsolicited message filters.



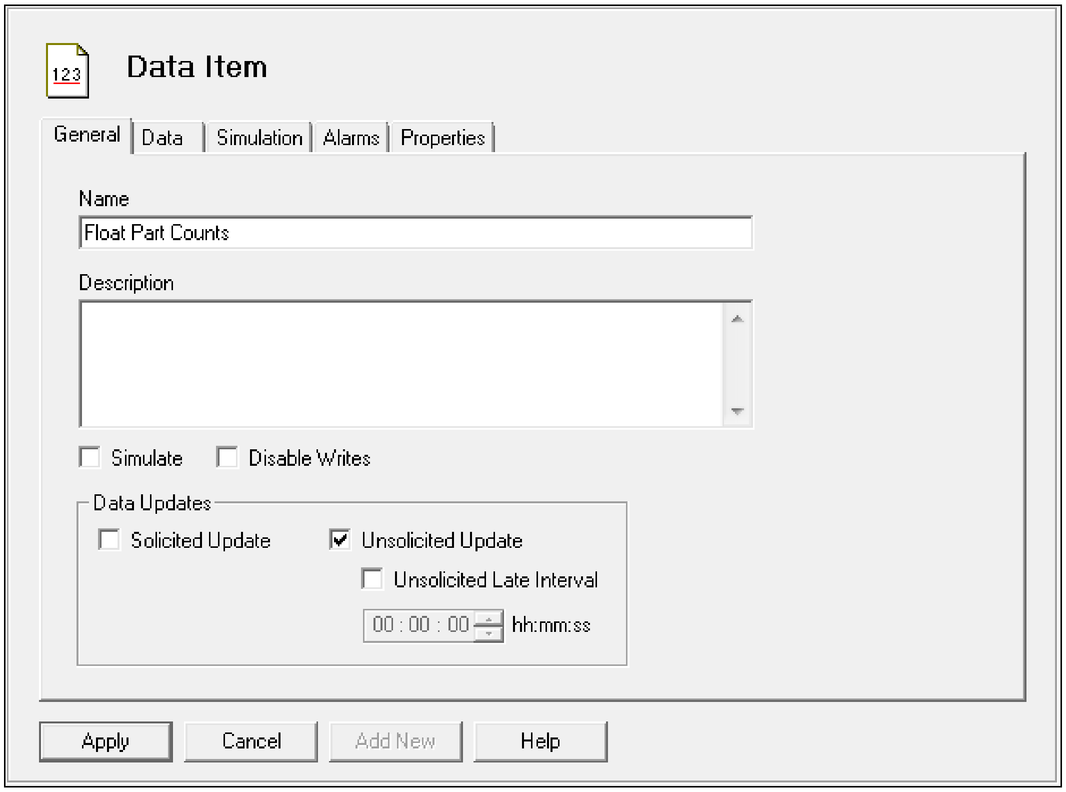
Groups	Network Connection	Network Node	DS Path Filter
☑ ⊟ Group1	(2 Message Filters)		
☑ Filter 1	Modbus Plus	OP40 Primary@2	1
☑ Filter 2	Modbus Plus	OP40 Backup@50	1

You will organize the filters in groups. There is no limit to the number of groups, and no limit to the number of filters within a group. Each group may be prioritized or non-prioritized.

If a group is non-prioritized, the server will accept messages that pass any filter in the group.

If a group is prioritized, the server will accept only those messages that pass the filter at the top of the list. If communication to that network node fails, then the server will accept only those messages that pass the next filter on the list, and so on.

There is no priority between groups. The server will accept a message that passes the filter criteria for any group.


Data Item

Name: Float Part Counts

Description:

Simulate [ ] Disable Writes [ ]

Data Updates

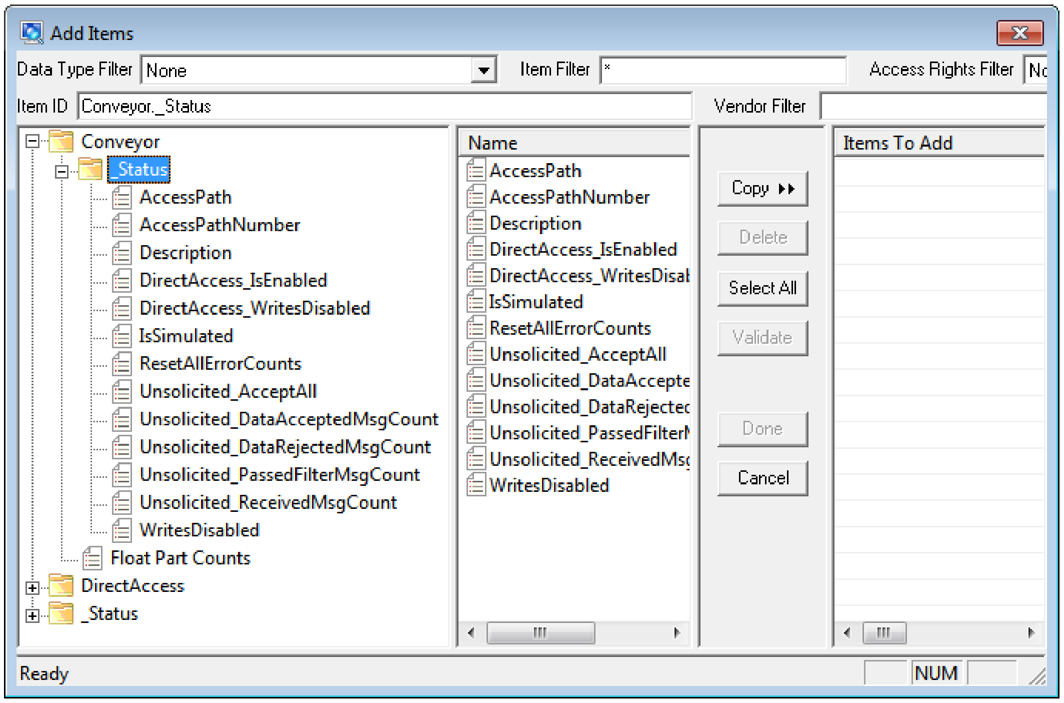
Solicited Update [ ] Unsolicited Update [✓]

Unsolicited Late Interval [ ]

00 : 00 : 00 hh:mm:ss

You must now enable unsolicited updates for each of the data items that will use that communication mode. To do this, simply check the ***Unsolicited Update*** box and uncheck the ***Solicited Update*** box on the data item's ***General*** tab.

You may also check the box for ***Unsolicited Late Interval***, and then specify an interval. If the controller fails to update the data item within this interval, the server will downgrade the data quality to Uncertain.



Data Type Filter: None
Item Filter: *
Access Rights Filter: No
Item ID: Conveyor._Status
Vendor Filter:

- Conveyor
  - _Status
    - AccessPath
    - AccessPathNumber
    - Description
    - DirectAccess_IsEnabled
    - DirectAccess_WritesDisabled
    - IsSimulated
    - ResetAllErrorCounts
    - Unsolicited_AcceptAll
    - Unsolicited_DataAcceptedMsgCount
    - Unsolicited_DataRejectedMsgCount
    - Unsolicited_PassedFilterMsgCount
    - Unsolicited_ReceivedMsgCount
    - WritesDisabled
  - Float Part Counts
- DirectAccess
- _Status

Unsolicited status items are available to OPC client applications.

The OPC server provides status data items for each device you create. Any OPC client can read these items to provide you information that may help in testing and debugging your configuration. Five of these status items relate to unsolicited communication:

Unsolicited_AcceptAll

This item reflects the state of the Accept All Unsolicited checkbox.

Unsolicited_DataAcceptedMsgCount

The number of unsolicited messages that delivered data to at least one data item under this device.

Unsolicited_DataRejectedMsgCount

The number of unsolicited messages that passed through the unsolicited filter, but could not deliver data to any data items under this device.

Unsolicited_PassedFilterMsgCount

The number of unsolicited messages that passed through the unsolicited filter associated with this device.

Unsolicited_ReceivedMsgCount

The number of unsolicited messages that this device received.

Caution! Unsolicited status items are created automatically by the OPC server, but only if there is at least one unsolicited filter group configured for the device. If you want to use these status items but you have not yet configured the unsolicited message filters (or if you do not plan to use filters), simply create an empty filter group for the device.

Technical Support

If you have any questions or problems with these procedures, please contact Cyberlogic's Technical Support group by emailing techsupport@cyberlogic.com, or by calling 248-631-2288.