Technical Note

Optimizing KEPServerEX V6 Projects

December 2020
Ref. 1.002

©2020 PTC, Inc. All Rights Reserved.

Table of Contents

1. Overview

Users can optimize server performance regardless of using an OPC client or a native interface for connection. The following topics are discussed in this document:

- Issues that affect performance and suggestions for resolution
- How communications occur between the server and device
- How the project's configuration affects the way the device communicates

2. Factors that Affect Communication Speed

The following factors affect the speed of device data updates:

1. The connection's bandwidth, which is speed at which the server is asked to communicate with the attached devices.
2. The wire time, which is the amount of time it takes for the server to transmit a request to the device, and the time it takes for the device to transmit a response back to the server.
3. The Device Turnaround Time, which is the amount of time it takes for the device to process an incoming request during normal communications.
4. The amount of time it takes for the device to process the request, as well as the percentage of total processor time that is devoted to communications.
5. The percentage of the device's Writes to Reads.
6. The configuration of the device's Retry and Timeout settings.

When a customer reports speed issues, Technical Support often asks the following questions to discuss project optimization.

- What is the connection's bandwidth?
- What is the number of devices on a single drop?
- What is the quality of the signal on the drop?
- What is the specified timeout rate and the number of retries?
- What is the rate at which data is scanned or requested? What is the rate in relation to the connection speed?
- What is the amount of data being requested, with regards to the rate at which data is being scanned and the speed of connection?
- Does the device have data block reads? If so, what size are they?
- How is the data that is being requested organized with regards to both data blocking and memory mapping?

2.1 Defining Bandwidth

Bandwidth is a connection's bit per second transmission capability. Although it is one of the most important factors of communication speed, it is also the least controllable. It is important to remember that without the involvement of any communications protocol, a serial connection of 9600 baud takes approximately 1 millisecond to transmit 1 byte. Therefore, if the device were transmitting 100 Words, it would take 200 milliseconds (or a maximum of five 100 Word transmissions in one second).

Although this sounds agreeable, most communications protocols add at least 2 bytes for start and stop characters, a byte for the Device ID, and a byte for the command. Thus, it is likely that an additional 8-10 bytes are added to the data request and response. Instead of 200 milliseconds, it will likely take a total of 230-260 milliseconds to send the request and receive the data.

The table below displays approximate data transfer rates. The Ethernet baud rates include the approximate number of bytes that can travel on the wire in 1 millisecond. They also include the number of milliseconds it takes to transfer one hundred 16-bit words (or 200 bytes).

Baud Rate	**Bytes/ 1 Millisecond**	**Milliseconds/100 Words**
300	.03125	6400
600	.0625	3200
1200	.125	1600
2400	.25	800
4800	.5	400
9600	1	200
19200	2	100
38400	4	50
57600	6	33.3
115200	12	16.6
10 MB	1041.67	.19
100 MB	10416.67	.0192

2.1.1 Factors that Slow Data Transfer

Many factors can negatively impact data throughput in cases where it is expected to perform quickly. For example:

- When talking to an Ethernet device over a dedicated modem line, the communication rate is limited to the bandwidth of that dedicated line. Therefore, even if the network bandwidth is 100 mb, some lines may be 57k baud or slower. The data is limited to the slower rate. This is evident in any media that narrows the expected bandwidth to the medium of connection.
- Terminal servers or Serial-to-Ethernet converters can have high bandwidth connectivity because of the Ethernet connection. Although users assume that this will be the rate of transfer, the rate of transfer for the serial connection to the device may be much slower than the Ethernet connection.

- The distance between protocol converters (which convert one protocol to another) also affects the time of data transfer. The farther a converter is from the other, the longer it will take. For instance, the protocols of a Modbus TCP to Modbus Serial converter are similar, so the conversion will be quick. If a Modbus TCP to Allen Bradley DF1 converter were used, the conversion would take longer because the mapping requirements are more extensive.

2.1.2 Radio Modems

Although most radio modems have a high bandwidth, they vary in degrees of impact. Some require that the server pause between receiving and sending data so that the radio may switch between send and receive; others only allow one connection. This forces the server to talk to the devices synchronously (and adds time to the poll cycle).

For more information, see Optimizing the Server Project.

2.2 Device Turnaround Time

The device turnaround time specifies the amount of time it takes the device to process a request and send it back. This can vary widely from device to device. Devices process cyclically: the inputs are read, the ladder is processed, the outputs are written, the communications are processed, and then the process repeats itself. There are two factors which affect device turnaround time: the amount of time dedicated to processing and the size of the project.

The amount of time that a device dedicates to processing communications or to its programming depends on both the manufacturer and the device model. Some devices (such as Allen Bradley ControlLogix) provide users the ability to adjust processor time percentages. Other devices use sub-function blocks so that only a small portion of the program will be called every processor scan cycle.

The size of the project will also impact how long it takes to process. The larger the project, the longer the process will take to complete. In such cases, little can be done to change the device turnaround time.

2.3 What is the Device Protocol?

The device protocol defines the format of read and write requests; specifically, the maximum size of the packet. Almost all device protocols have small packet sizes. Although reading and writing multiple data items from the device usually requires that the data to be in contiguous address, many devices allow data in noncontiguous data addresses to be requested in a single request. Devices usually respond faster if items are in contiguous addresses because it is easier and quicker to grab them. Drivers in the server are limited to what they can do by the manufacturers' communication protocol.

2.3.1 Blocking Data

Except for a few drivers, the server calculates the block offsets from the first register. For example, if the block size for holding registers in the Modbus Driver is set to the maximum of 120, the first block would be from register 1-120, the second from 121–240, and so on. The driver would take the requested items, determine where they fall within a block, and then specify the amount of registers to request at a time. The driver would not request data across a block boundary unless it was for an array or string.

If the project requests data from registers 1-8, 50, and 58-64, the driver would ask for registers 1-64; if the project requests 1-8 and 120, the driver would ask for 120 registers. It is faster to ask for data that is not needed than to ask for several small packets.

For more information, see What the Client Wants Vs. What the Client Gets.

2.4 Signal Quality

Ideally, a connection to the device is crystal clear; however, cables may run for several hundred feet to connect to a PLC (and may pass electrical equipment, electrical conduits, and lights) so an ideal connection is unlikely. Such connections may be made through radio modems, cellular modems, phones, local intranet, or internet, which can raise several problems:

- Electrical equipment (such as heaters and coolers) creates large phased fields around themselves that can block or garble a signal in the line.
- Sunspot activity and physical obstacles can interfere with radio modems.
- The bandwidth on a phone or internet connection may not be large enough for fast communications.

When the signal quality decreases, messages can become garbled and undecipherable. In many cases, no response will be received from the device. When this occurs, the server must retry the request (which can take several seconds).

2.5 Device Drops

Communications entail a synchronous process of requests and responses. Device drop refers to how many devices will be dropped off a single communications port or media converter, in addition to how many devices will be dropped from a single channel in the server. Channels represent a single communications port or source socket on the PC.

Note: The more devices on a drop, the longer it will take to poll each one.

2.6 Polling

The data polling rate depends on bandwidth, device turnaround time, signal quality, and device protocol. For example, a project configured to poll data from the device every 10 milliseconds may not accomplish that rate unless the following specifications are met:

- The data can be returned in a single packet
- The device can respond to the request in less than 10 milliseconds.

2.7 Example of Communications

A server is configured with three devices on the same channel. It takes 300 milliseconds to receive data from two devices, and 500 milliseconds to receive data from one. Therefore, the poll cycle may be represented as follows:

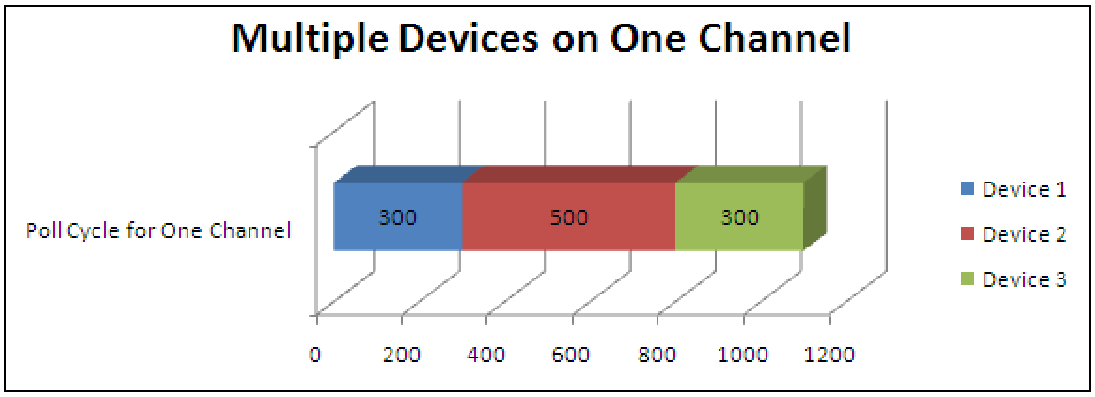


Image: Poll Cycle for Multiple Devices on a Channel

As shown in the image above, it takes a total of 1100 milliseconds to poll the data from all three devices at once. If communications were lost to Device 1, however, the poll cycle would increase.

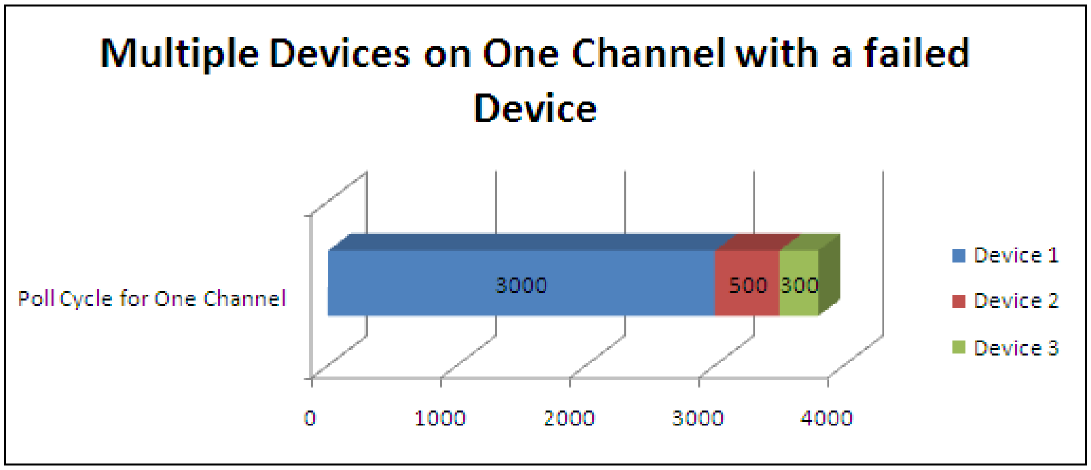


Image: Poll Cycle for Multiple Devices on a Channel with One Device Failing

As shown in the image above, the poll cycle has significantly increased to 3800 milliseconds. Information on avoiding or resolving this situation will be discussed later in Updating the Project Structure.

©2020 PTC, Inc. All Rights Reserved.

3. Optimizing the Server Project

Users can do several things in the server, the physical device, and the client application to optimize project performance.

3.1 Device Properties

The rate at which data is transmitted and received depends on the number of devices being connected to at one time. It is important to remember that bandwidth applies to the entire channel, not just to one device.

In Example of Communications, three devices were on the same line. Two of the devices each took 300 milliseconds to provide data, and one took 500 milliseconds to provide data: they combined for a total of 1100 milliseconds. The example assumed clear communications with the devices; however, in an industrial environment, the communications line will likely experience noise. In this case, the server would receive a garbled response or no response before eventually timing out and retrying the request.

To resolve this issue, users should separate the devices onto their own channels. Three separate communications threads allow the server to talk simultaneously to all three devices. By talking to each device separately, the poll cycle on each channel would be the rate that it takes to get the data from that device.

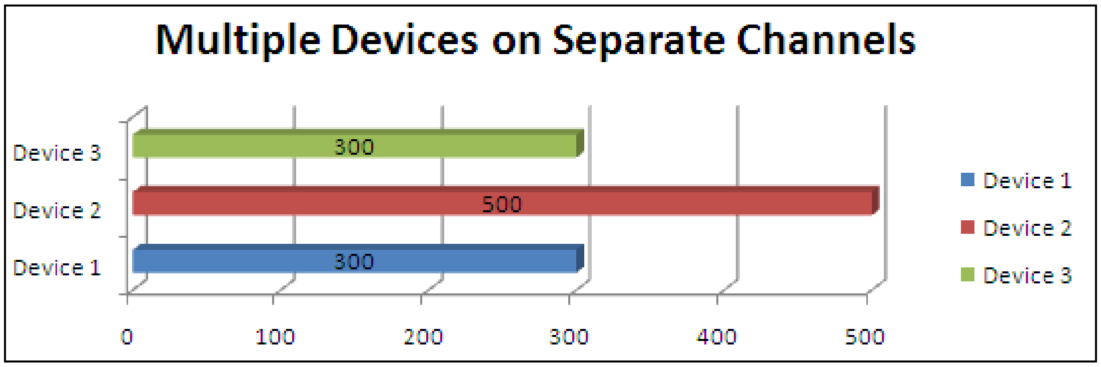
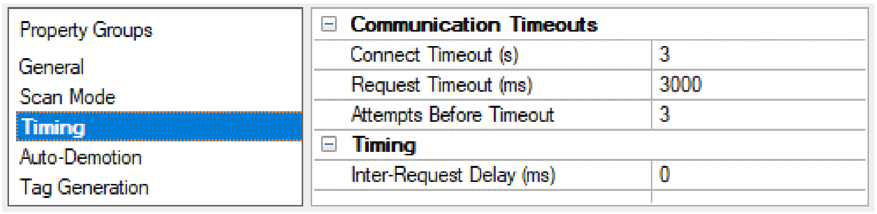


Image: Poll Cycle for Multiple Devices on Separate Channels

Note: Some device protocols do not allow multiple channels from the same source IP address. In such cases, the PC's network card must be multi-homed in order to use multiple channels. *For more information, refer to the driver's help documentation*.

3.1.1 Device Timing Settings

Although most users utilize the default communication settings, the parameters may be changed by clicking **Device Properties** | **Timing**.



Communication Timeouts	
Connect Timeout (s)	3
Request Timeout (ms)	3000
Attempts Before Timeout	3
Timing	
Inter-Request Delay (ms)	0

Image: Timing Property Group

Descriptions of the properties are as follows:

- **Connect timeout (s):** This property (which is used primarily by Ethernet based drivers) controls the amount of time required to establish a socket connection to a remote device. The device's connection time often takes longer than normal communications requests to that same device. The valid range is 1 to 30 seconds. The default is typically 3 seconds but can vary depending on the driver's specific nature. If this setting is not supported by the driver, it is disabled.

  Note: Due to the nature of UDP connections, the connection timeout setting is not applicable when communicating via UDP.

- **Request timeout (ms):** This property specifies an interval used by all drivers to determine how long the driver waits for a response from the target device to complete. The valid range is 50 to 9,999,999 milliseconds (167.6667 minutes). The default is usually 1000 milliseconds but can vary depending on the driver. The default timeout for most serial drivers is based on a baud rate of 9600 baud or better. When using a driver at lower baud rates, increase the timeout to compensate for the increased time required to acquire data.

- **Attempts Before Timeout:** This property specifies how many times the driver issues a communications request before considering the request to have failed and the device to be in error. The valid range is 1 to 10. The default is typically 3 but can vary depending on the driver's specific nature. The number of attempts configured for an application depends largely on the communications environment. This property applies to both connection attempts and request attempts.

- **Inter-Request Delay (ms):** This property specifies how long the driver waits before sending the next request to the target device. It overrides the normal polling frequency of tags associated with the device, as well as one-time reads and writes. This delay can be useful when dealing with devices with slow turnaround times and in cases where network load is a concern. Configuring a delay for a device affects communications with all other devices on the channel. It is recommended that users separate any device that requires an inter-request delay to a separate channel if possible. Other communications properties (such as communication serialization) can extend this delay. The valid range is 0 to 300,000 milliseconds; however, some drivers may limit the maximum value due to a function of their particular design. The default is 0, which indicates no delay between requests with the target device.

  Note: Not all drivers support Inter-Request Delay. This setting does not appear if it is not available.

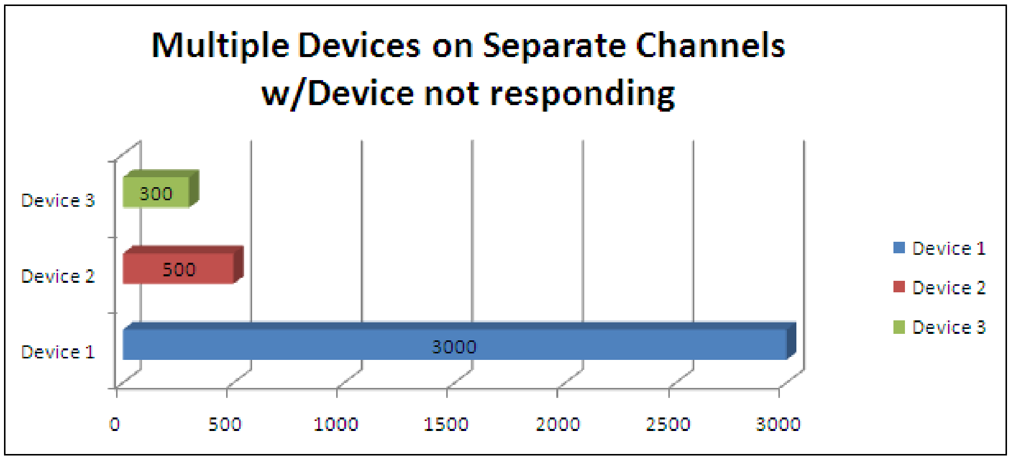
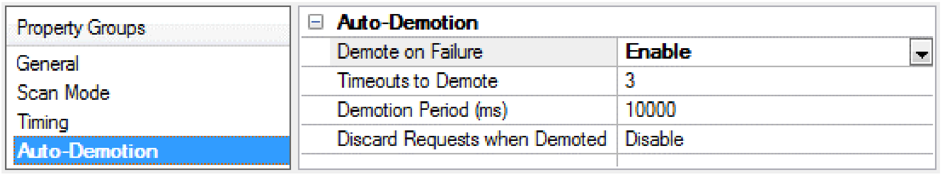


Image: Poll Cycle for Multiple Devices on Separate Channels

Note: To make more attempts with less time to report a communications failure, configure a faster timeout with more retries. For example, a timeout of 500 milliseconds with four attempts adds an additional attempt for a response to a request. It also allows for complete failure of communications to be reported in two seconds instead of three.

3.1.2 Auto-Demotion

A project can usually be optimized by simply placing the devices on individual channels so they can communicate simultaneously. In some situations, devices must be dropped from a single connection (such as for some radio modems or serial to Ethernet converters). In these cases, auto-demotion allows the driver to automatically demote a device from the poll cycle if it becomes unresponsive.



Auto-Demotion	
Demote on Failure	Enable
Timeouts to Demote	3
Demotion Period (ms)	10000
Discard Requests when Demoted	Disable

Image: Auto-Demotion Property Group

Descriptions of the properties are as follows:

- **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 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**.**

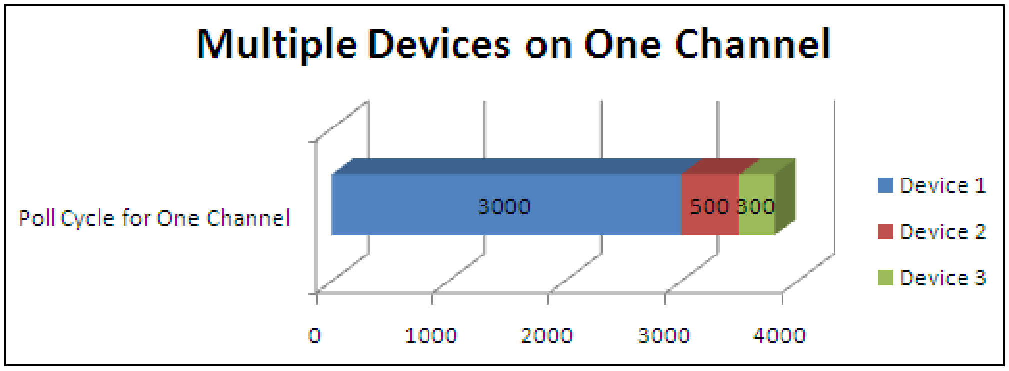


Image: Chart of Poll Cycle with Failed Device

Note: When Auto-Demotion is enabled, the device will be removed from the poll cycle. It will not impact the polling of the rest of the devices.

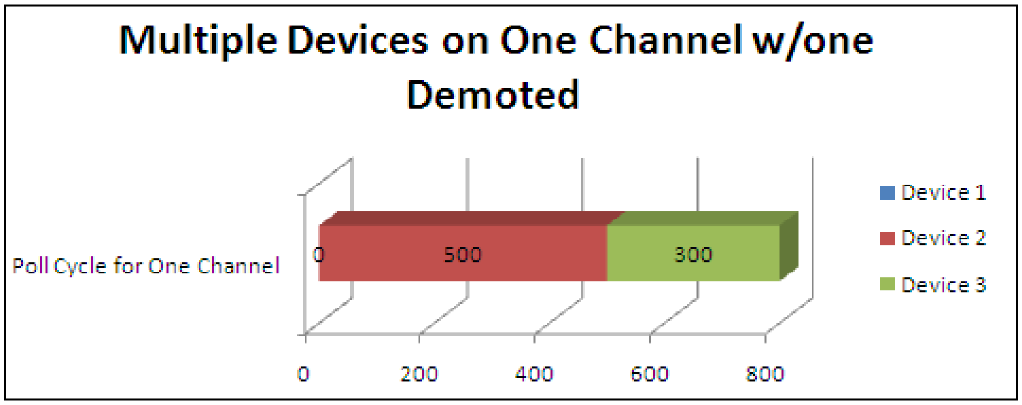


Image: Chart of Poll Cycle with Demoted Device

Note: Users receiving frequent device demotions should attempt to identify and correct the cause.

3.2 The Alias and Alias Map

The alias and the Alias Map originally intended to provide simple mapping for DDE application topics. This was especially useful for customers that were porting from legacy KEPServerEX versions.

An alias can be used with OPC Clients to optimize the project structure without changing its configuration. This is very useful for clients that do not provide a means to globally modify item databases.

3.2.1 Updating the Project Structure

Although configuring a project with all devices on a single channel can slow the processing time, it allows the server to talk to all devices simultaneously. This decreases the size of the poll cycle and ensures that one device's communications issues will not affect another's.

Note: An alias cannot share a name with a channel and device combination.

The following instructions describe how to create an alias that is the same name as the existing channel and device.

1. To start, create three channels with one device each.

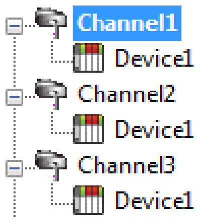
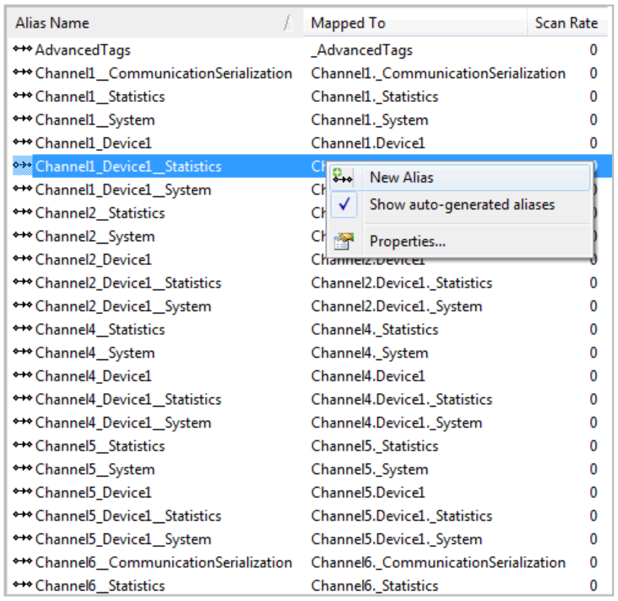


Image: An Optimized Project

2. Select **Alias** in the tree view, then in the server's main toolbar click **Edit** | **Aliases** | **Show auto-generated aliases**. An alias will be automatically created for each location in the project where item tags are located. The alias name is the original channel and device name separated by a period. For example, "Channel1.Device1".

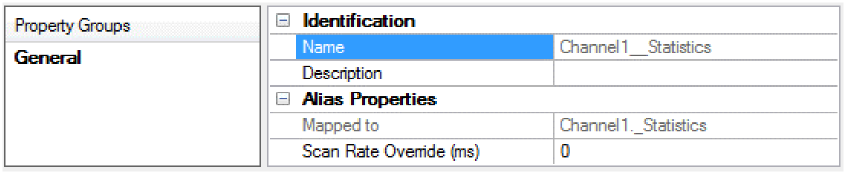
Note: The server will replace all periods with an underscore. This is done for DDE clients that do not allow periods in DDE topics.



Alias Name	Mapped To	Scan Rate
AdvancedTags	_AdvancedTags	0
Channel1__CommunicationSerialization	Channel1._CommunicationSerialization	0
Channel1__Statistics	Channel1._Statistics	0
Channel1__System	Channel1._System	0
Channel1_Device1	Channel1.Device1	0
Channel1_Device1__Statistics	Ch	
Channel1_Device1__System	Ch	
Channel2__Statistics	Ch	
Channel2__System	Ch	
Channel2_Device1	Channel2.Device1	0
Channel2_Device1__Statistics	Channel2.Device1._Statistics	0
Channel2_Device1__System	Channel2.Device1._System	0
Channel4__Statistics	Channel4._Statistics	0
Channel4__System	Channel4._System	0
Channel4_Device1	Channel4.Device1	0
Channel4_Device1__Statistics	Channel4.Device1._Statistics	0
Channel4_Device1__System	Channel4.Device1._System	0
Channel5__Statistics	Channel5._Statistics	0
Channel5__System	Channel5._System	0
Channel5_Device1	Channel5.Device1	0
Channel5_Device1__Statistics	Channel5.Device1._Statistics	0
Channel5_Device1__System	Channel5.Device1._System	0
Channel6__CommunicationSerialization	Channel6._CommunicationSerialization	0
Channel6__Statistics	Channel6._Statistics	0

Image: Alias Map with Automatically Defined Aliases

3. To create a new alias, right-click the Detail View and select **New Alias** (or select **Edit** | **Aliases** | **New Alias**).
4. Browse to the group or device that contains the item to be referenced and enter the original channel and device as the name.



Property Groups		
General	**Identification**	
	Name	Channel1__Statistics
	Description	
	Alias Properties	
	Mapped to	Channel1._Statistics
	Scan Rate Override (ms)	0

Image: Alias Properties

5. Repeat the process for all devices that require optimized communications.

Note: Each alias has a scan rate override that can be used to override the Tag Scan rate that is provided for non-OPC Clients. For more information, refer to What the Client Wants vs. What the Client Gets.

6. When finished, click **OK**. The new aliases can be found in the Alias Map.

3.2.2 How the Alias Works with an OPC Client

When a client application adds an item to the server, the server will first check to make sure it is valid. If the client adds item "E1.D1.Tag1," the server will look for tag on channel "E1" and device "D1." When it does not find the channel and device, it will check to see if there is an alias with the name "E1.D1." Then, it will check the mapped device to see if there is a tag with that name. At this point, the client has no idea that it is connected through an alias.

Note: An alias is not needed for each tag group under a device. The server knows that it only needs to match part of the alias.

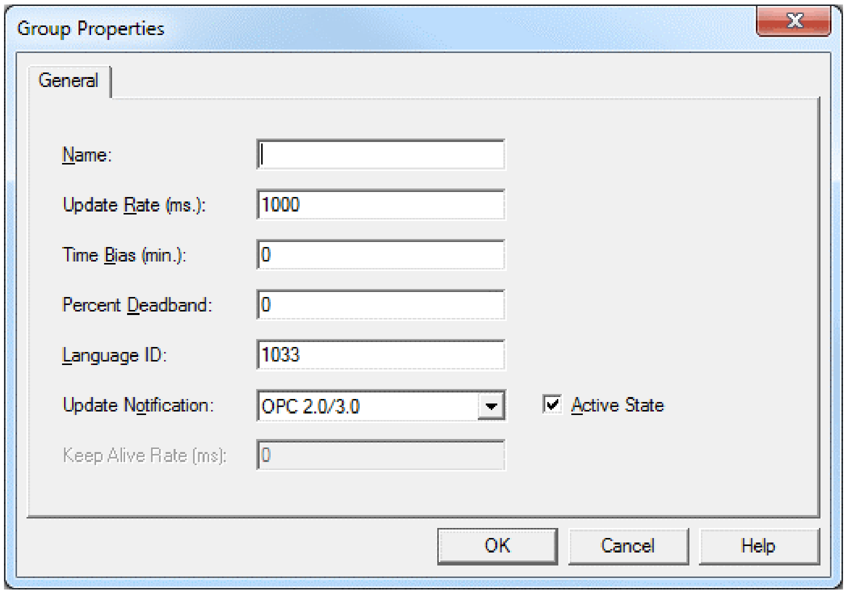
3.3 What the Client Wants vs. What the Client Gets

The client does not always get what it wants. When requesting data from the server, it is important to consider the bandwidth, the amount of requested data, the device turnaround time, and the configured timeout settings. Each of these settings may be adjusted to optimize the data collection process. OPC Clients dictate the way that the server receives data and specify the rate at which it will be polled.

A common issue occurs when data changes at a 300 milliseconds rate, but it takes at least 300 milliseconds to poll the device for the data. In such instances, there is no guarantee that users will receive the changing data each poll cycle.

3.3.1 Client Update Rates

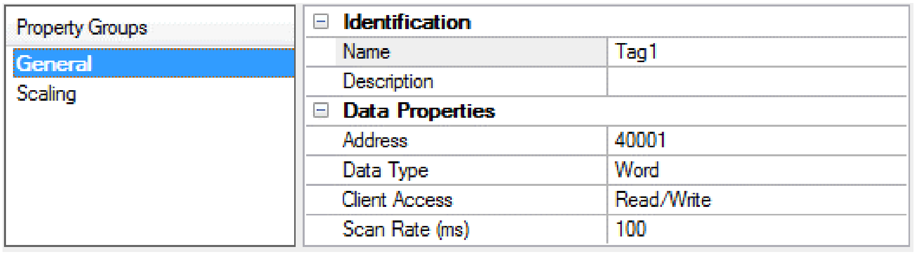
Since Kepware products support OPC, DDE, and other data transfer methods, it's important to note the type of connection for the client in order to properly optimize the poll rates. OPC clients pass the update rate as part of an OPC group property. This property can be then adjusted to the rate at which items in the group are requested by the device. OPC clients can also have OPC groups with different update rates – this allows faster polling for critical data than static data (such as set points).



Name:	
Update Rate (ms.):	1000
Time Bias (min.):	0
Percent Deadband:	0
Language ID:	1033
Update Notification:	OPC 2.0/3.0
☑ Active State	
Keep Alive Rate (ms):	0

Image: OPC Quick Client General Group Properties

For non-OPC Clients, a scan rate is associated with each tag. This scan rate can only change after the client has released the item, then the new scan rate is applied to the next client request. Different scan rates can be assigned to each item to allow for disparities between the polling requirements of critical and static data.



Identification	
Name	Tag1
Description	
Data Properties	
Address	40001
Data Type	Word
Client Access	Read/Write
Scan Rate (ms)	100

Image: Tag General Property Group

Note: To override the tag scan rate for DDE-type client connections, set a scan rate override on an alias. *For more information, refer to the driver* *help documentation*.

3.3.2 Using the Scan Rate and the Device Protocol for Optimization

When designing a project for the first time or modifying an existing project, users can optimize the project by placing all the critical data in one or more consecutive blocks. In the client application, all critical data should be placed in its own OPC group. This allows the OPC client to set a faster poll rate for critical data and allows the driver to optimize the data requests.

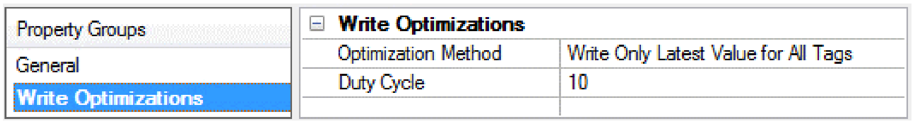
3.4 Managing Writes

When a client writes data to a device, the server will stop polling the device to accomplish the write. It is treated as an immediate action. The server writes one item at a time, and additional writes are placed into a queue and processed as soon as possible. Without some sort of management, users could potentially flood the device with writes and never get any reads.

Note: Array tags are considered single items.

3.4.1 Optimizing the Write Process

Each channel that is created in a server project has a Write Optimizations property page where users can specify the Optimization Method and Duty Cycle.



Write Optimizations

Optimization Method	Write Only Latest Value for All Tags
Duty Cycle	10

Image: Write Optimizations Property Group

By default, the Optimization Method is set to **Write Only Latest Value for All Tags**. This specifies that if an application is writing values every 10 seconds, and it is taking 300 milliseconds to complete a single write, then the server will replace a pending write value with a newer value instead of queuing 30 writes to the same tag. Since there are times when multiple writes to the same tag is required, there are options to **Write Only Last Value for Non-Boolean Tags** or **Write Only Latest Values for All Tags**.

The Duty Cycle specifies the number of writes before performing a read transaction (when there are outstanding writes in the write queue). The default setting is 10.

4. Summary

There are many factors involved in efficient communication. Keep these factors in mind to apply the best optimization technique and plan for the best performance.