# Kepware Technologies

# Optimizing KEPServerEX V5 Projects

September, 2010
Ref. 50.16

# Table of Contents

# Table of Figures

# 1. Overview

Users must understand how to optimize server performance regardless of using an OPC client or a native interface for connection. As such, the following topics will be 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 in order 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 have been added to the data request and response, as well. 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 perform quickly. Examples are as follows:

- 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 57 k 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 is usually 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 example, since the protocols of a Modbus TCP to Modbus Serial converter are similar, the conversion will be quick. If a Modbus TCP to Allen Bradley DF1 converter were used, however, 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, refer to 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. Two factors in particular 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. Unfortunately, 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, however, because it is easier and quicker to grab them. Thus, the drivers in the server are limited to what they can do by the manufacturers' communication protocol.

### *2.3.1 Blocking Data*

With the exception of 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, then 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 how 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, refer to What the Client Wants Vs. What the Client Gets.

## 2.4 Signal Quality

An ideal connection to the device is crystal clear; however, since cables may run for several hundred feet in order to connect to a PLC (and may pass electrical equipment, electrical conduits, and lights) the ideal connection is not likely. The connection itself may be made through radio modems, cellular modems, phones, local intranet, or internet. As such, there are several factors that raise potential problems:

- Electrical equipment (such as heaters and coolers) creates large phased fields around themselves that may block or garble a signal in the line.
- Sunspot activity and physical obstacles may 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 off of 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 the 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 has the ability to 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:

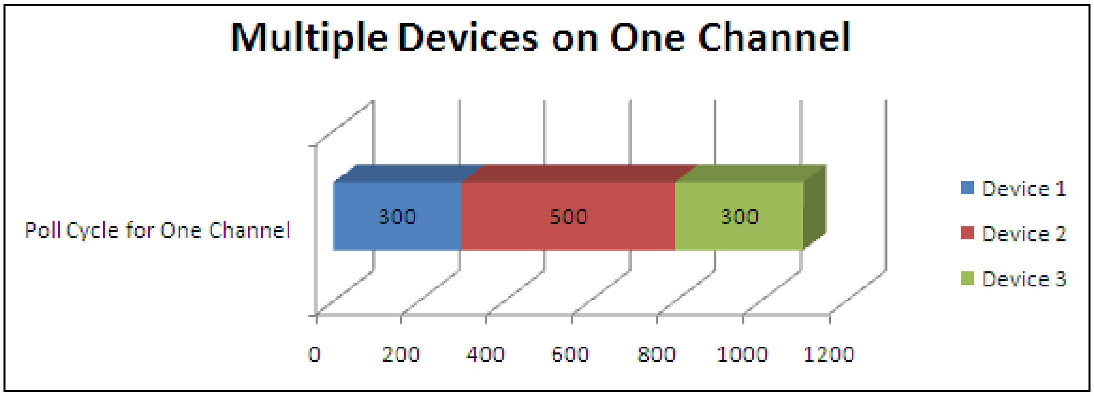


**Figure 1: Poll Cycle for Multiple Devices on a Channel**

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.

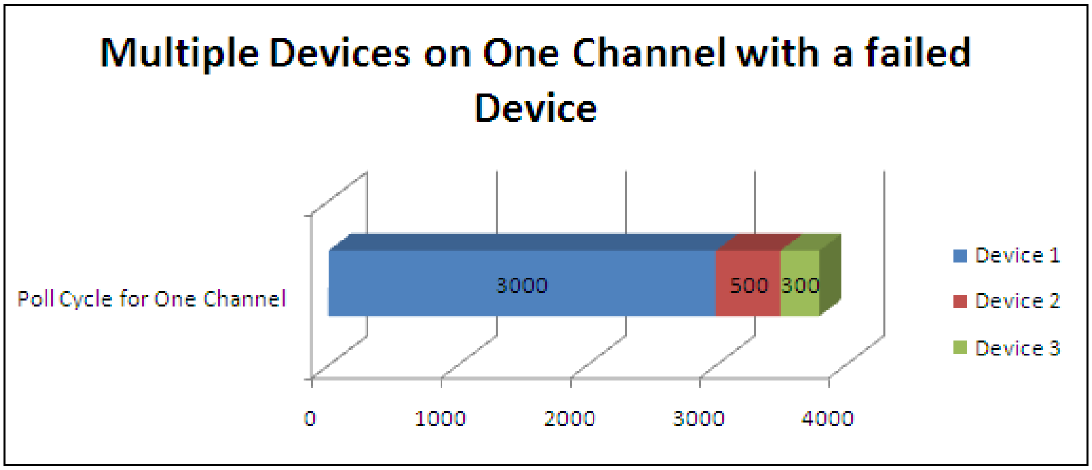


**Figure 2: Poll Cycle for Multiple Devices on a Channel with One Device Failing**

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 this documentation.

# 3. Optimizing the Server Project

Users can do several things in the server, the physical device, and the client application in order 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. Furthermore, users must 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.

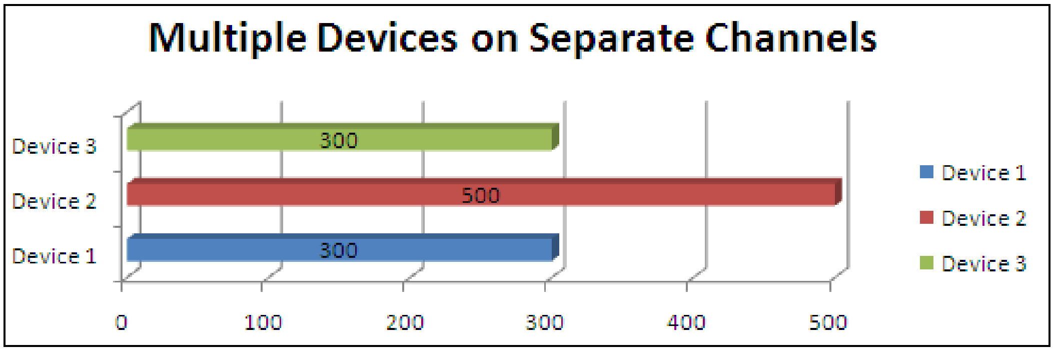
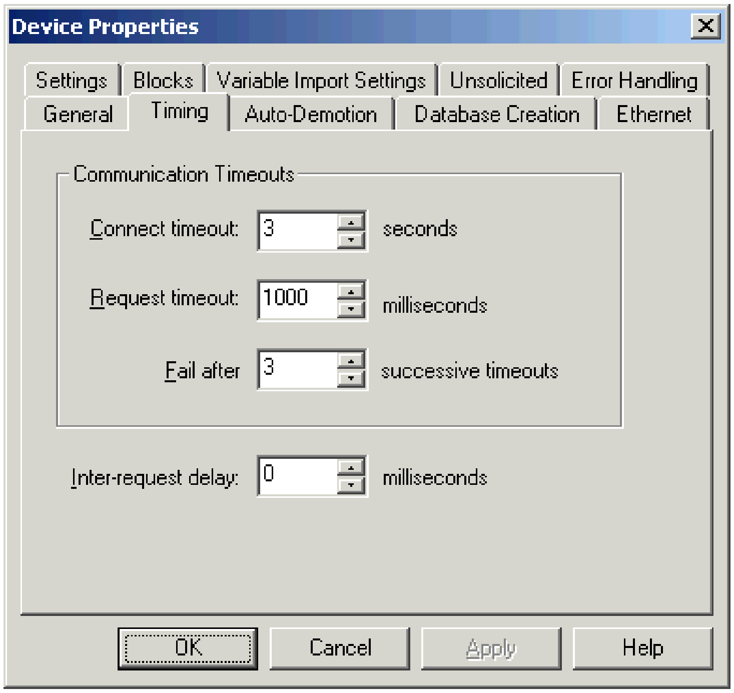


**Figure 3: Poll Cycle for Multiple Devices on Separate Channels**

**Note:** Some device protocols do not allow multiple channels from the same source IP address. Therefore, users must multi-home the PC's network card 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: 3 seconds

Request timeout: 1000 milliseconds

Fail after 3 successive timeouts

Inter-request delay: 0 milliseconds

**Figure 4: The Device Properties' Timing Dialog**

Descriptions of the parameters are as follows:

- **Connect timeout:** This parameter specifies how long the server will wait before failing on the initial connection to the device via an Ethernet connection. The default setting is 3 seconds.
- **Request timeout:** This parameter specifies the amount of time that the server will wait for a response before it resends the request. The default setting is 1000 milliseconds.

  **Note:** The request timeout should be adjusted to 1.5 times the slowest response that the device sends back for a request. For example, if most data is returned in 20 milliseconds but one request takes 100 milliseconds, users should set the request timeout to 150 milliseconds.
- **Fail after __ successive timeouts:** This parameter specifies the number of times that the request will be attempted before it fails. Once it fails, the device's communication error tag will be set to a triggered state. The default setting is 3 timeouts.
- **Inter-request delay:** This parameter specifies the amount of time that the server will pause before sending the next request. It is used for interfaces that need to reset (or switch from a state of receiving to sending). The default setting is 0 milliseconds. This parameter is not enabled on all drivers, and will be grayed out if not implemented.

  The inter-request delay is typically used with radio modems. The server is designed to process requests as fast as it can: multiple poll requests are sent to get data as soon as a response from one request is received. Then, the next will be sent.

  **Note:** Many radio modems require a few milliseconds to switch from receive to transmit.

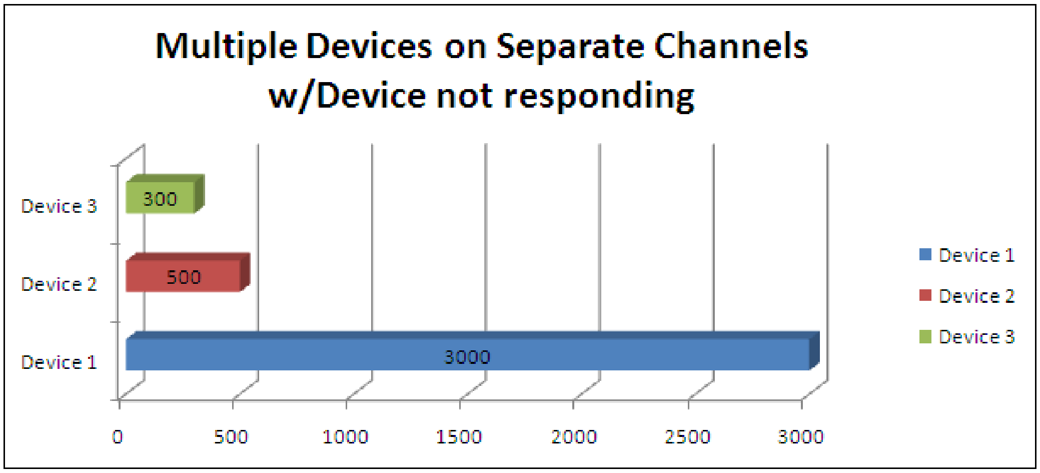
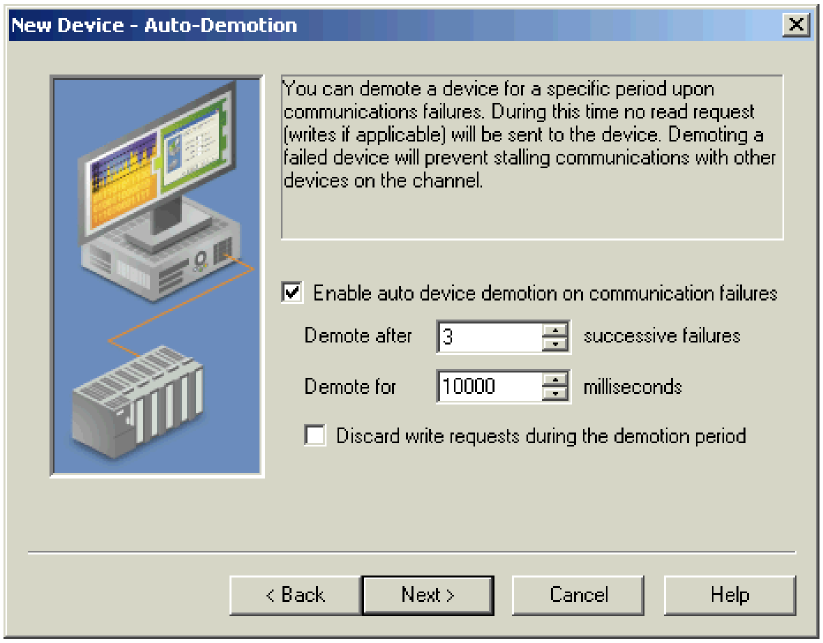


**Figure 5: Poll Cycle for Multiple Devices on Separate Channels**

**Note:** In order for more attempts to be made with less time to report a complete communications failure, users can 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 simply by placing the devices on individual channels so that they can communicate simultaneously. In some situations, devices must be dropped off of a single connection (such as for some radio modems or serial to Ethernet converters). In these cases, users have the option to automatically demote a device from the poll cycle if it becomes unresponsive. This is called auto-demotion.



New Device - Auto-Demotion

You can demote a device for a specific period upon communications failures. During this time no read request (writes if applicable) will be sent to the device. Demoting a failed device will prevent stalling communications with other devices on the channel.

☑ Enable auto device demotion on communication failures

Demote after 3 successive failures

Demote for 10000 milliseconds

□ Discard write requests during the demotion period

**Figure 6: Auto-Demotion**

Descriptions of the parameters are as follows:

- **Enable auto device demotion on communications failures:** When checked, auto-demotion will be enabled for communications failures. The default setting is disabled.
- **Demote after __ successive failures:** This parameter defines the number of successive failures that can occur before the device will be demoted. The default setting is 3.
- **Demote for __ milliseconds:** This parameter defines how long the device should be demoted before communications can be reattempted. The default setting is 10000 milliseconds.
- **Discard write requests during the demotion period:** When checked, write requests will be discarded when a device is demoted. This setting is important: writes can be made to a demoted device and will continue to queue for processing. In a production environment, it could be dangerous to have a process start mid-step because of a queued write. By discarding those writes, users ensure that an action is not taken inadvertently.

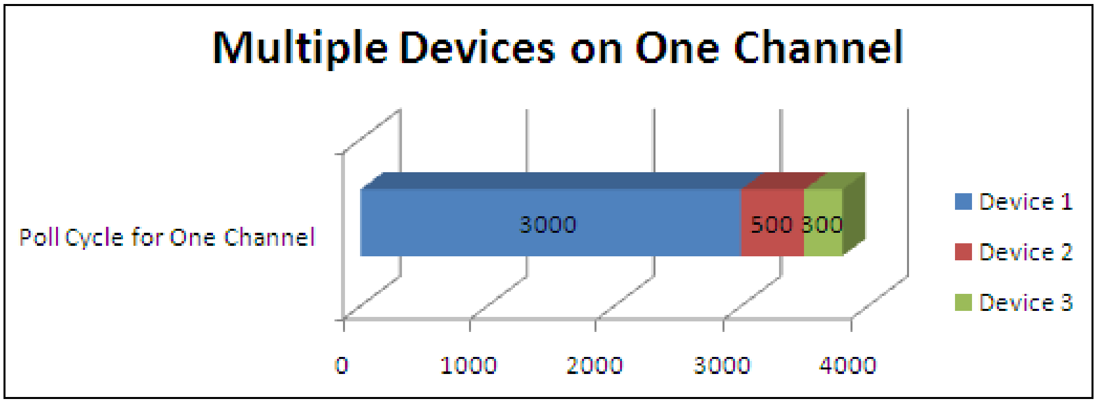


**Figure 7: Chart of Poll Cycle with Failed Device**

**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.

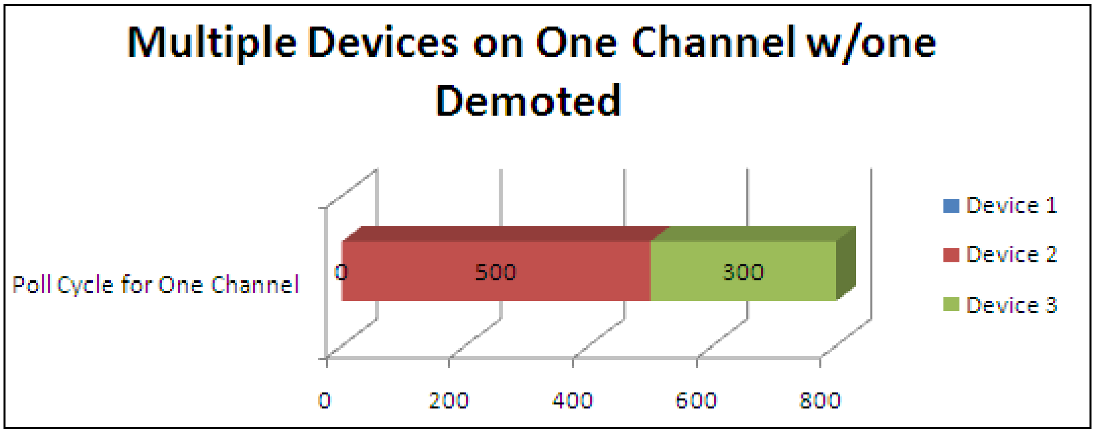


**Figure 8: Chart of Poll Cycle with Demoted Device**

**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 KEPServer 3.2 to KEPServerEX V4. It has since been carried over to KEPServerEX V5.

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 cannot always be avoided. By placing the device on its own channel, however, the server can talk to all devices simultaneously. This decreases the size of the poll cycle, and also ensures that one device's communications issues will not affect another's.

**Note:** The 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 exiting channel and device.

1. To start, create three channels.
2. Create a device under each channel. In this example, E2 is with D1, E3 is with D2, and E4 is with D3.

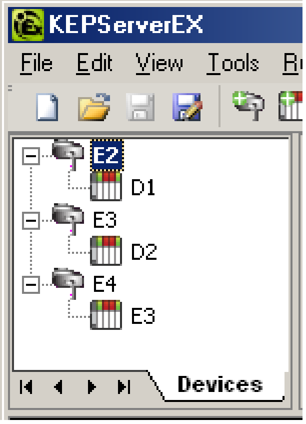
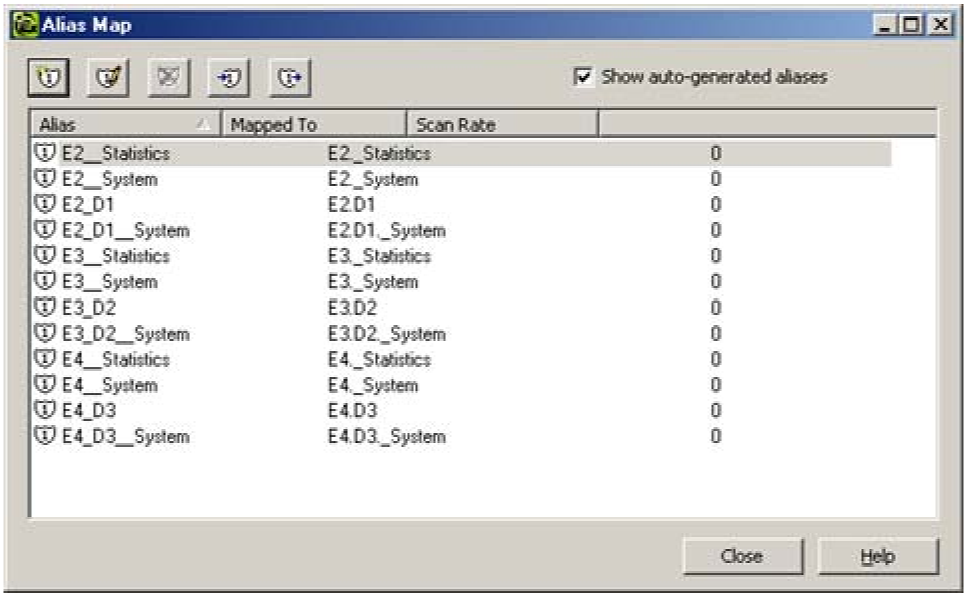


**Figure 9: An Optimized Project**

3. Next, click **Edit** | **Alias Map…** in the server's main toolbar. An alias will be automatically created for each location in the project where item tags are located.

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



☑ Show auto-generated aliases

| Alias | Mapped To | Scan Rate |
|---|---|---|
| E2__Statistics | E2._Statistics | 0 |
| E2__System | E2._System | 0 |
| E2_D1 | E2.D1 | 0 |
| E2_D1__System | E2.D1._System | 0 |
| E3__Statistics | E3._Statistics | 0 |
| E3__System | E3._System | 0 |
| E3_D2 | E3.D2 | 0 |
| E3_D2__System | E3.D2._System | 0 |
| E4__Statistics | E4._Statistics | 0 |
| E4__System | E4._System | 0 |
| E4_D3 | E4.D3 | 0 |
| E4_D3__System | E4.D3._System | 0 |

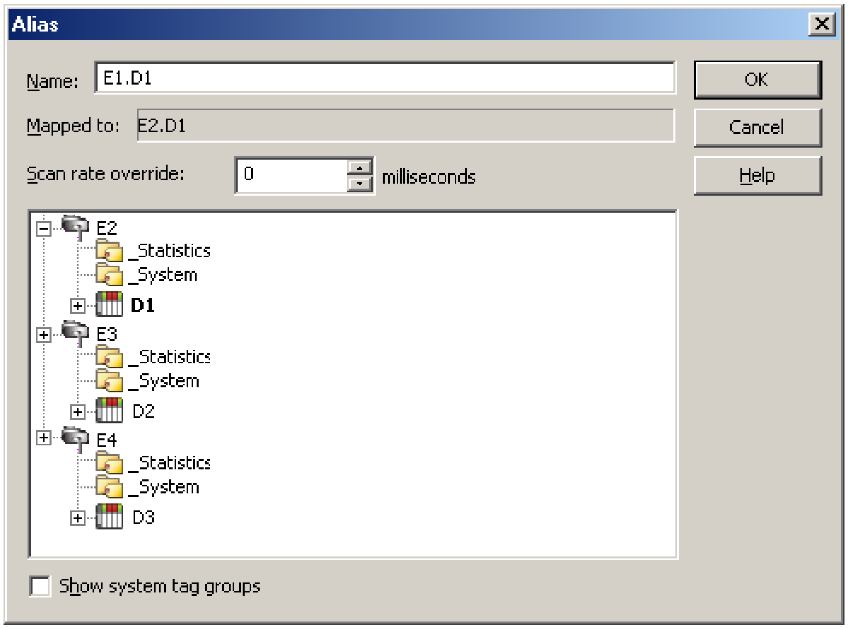
**Figure 10: Alias Map with Automatically Defined Aliases**

4. Next, click **New Alias** to open the **Alias Property** dialog.
5. Select the device for which the alias will be created, and then enter the original channel and device as the name.

   **Note:** In this example, D1 on E2 is being selected and assigned the alias name "E1.D1."

6. 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.



Name: E1.D1

Mapped to: E2.D1

Scan rate override: 0 milliseconds

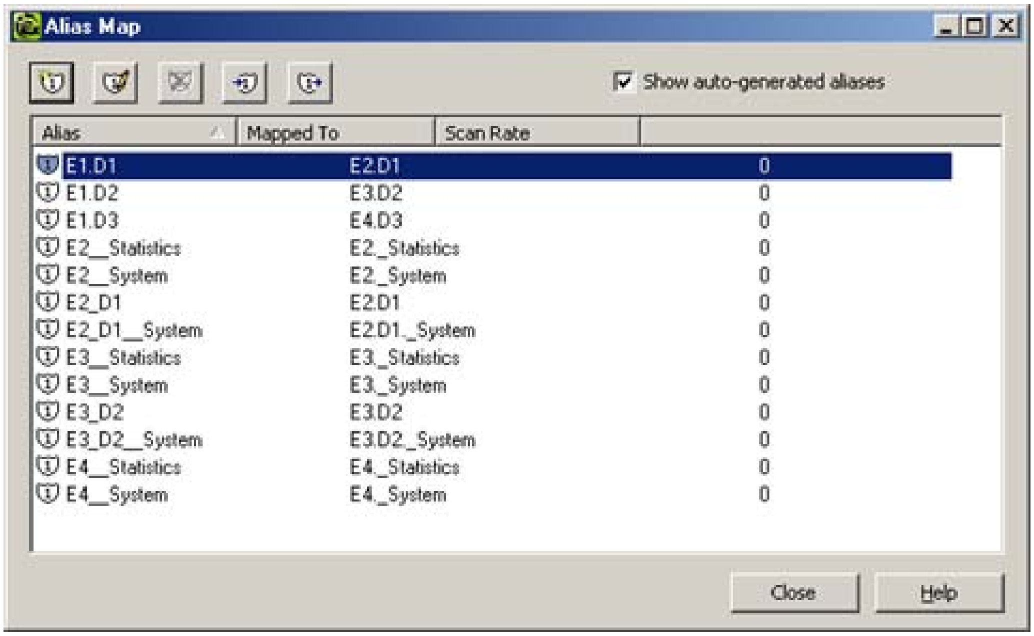
```
E2
  _Statistics
  _System
  D1
E3
  _Statistics
  _System
  D2
E4
  _Statistics
  _System
  D3
```

□ Show system tag groups

**Figure 11: Alias**

7. When finished, click **OK**.

   **Note:** Upon completion, the new aliases will be listed in the Alias Map.



☑ Show auto-generated aliases

| Alias | Mapped To | Scan Rate |
|---|---|---|
| E1.D1 | E2.D1 | 0 |
| E1.D2 | E3.D2 | 0 |
| E1.D3 | E4.D3 | 0 |
| E2__Statistics | E2._Statistics | 0 |
| E2__System | E2._System | 0 |
| E2_D1 | E2.D1 | 0 |
| E2_D1__System | E2.D1._System | 0 |
| E3__Statistics | E3._Statistics | 0 |
| E3__System | E3._System | 0 |
| E3_D2 | E3.D2 | 0 |
| E3_D2__System | E3.D2._System | 0 |
| E4__Statistics | E4._Statistics | 0 |
| E4__System | E4._System | 0 |

**Figure 12: Alias Map with User-Defined Aliases**

### *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.

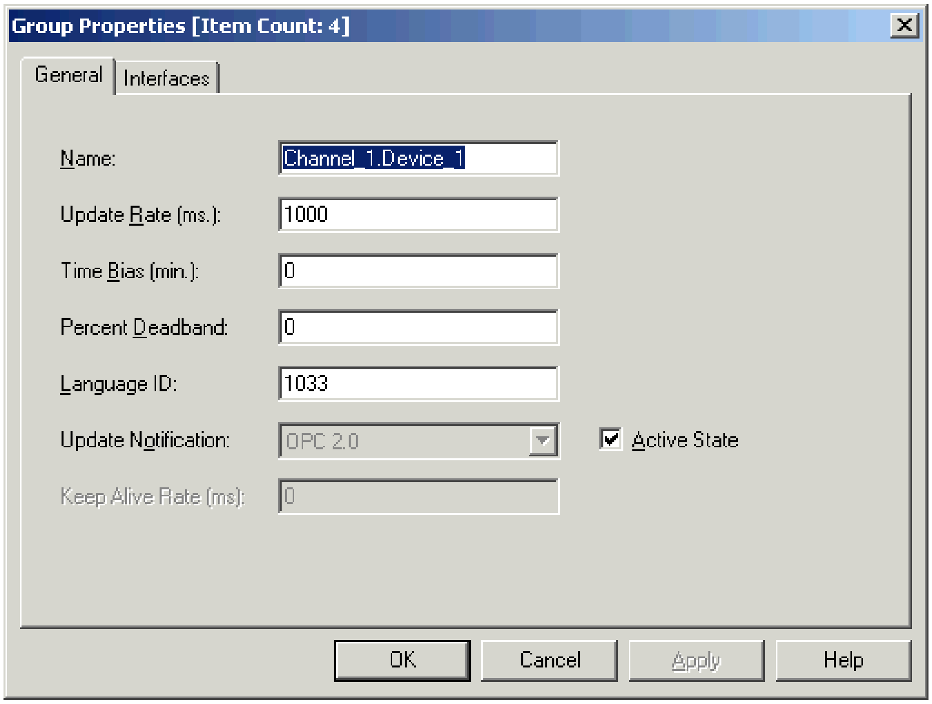
## 3.3 What the Client Wants Vs. What the Client Gets

The client does not always get what it wants. When requesting data from the server, users must 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 also specify the rate at which it will be polled.

Many users encounter a problem wherein they may have data changing at a 300 milliseconds rate, but it takes at least 300 milliseconds to poll the device for the data once. There is no guarantee that users will receive the changing data each poll cycle.

### *3.3.1 Client Update Rates*

Since Kepware Technologies products support OPC, DDE, and other data transfer methods, users should be aware of the type of connection being made by the client in order to properly optimize the poll rates. OPC Clients will 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 being requested by the device. OPC Clients can also have OPC groups with different update rates: this allows critical data to be polled faster than static data (such as set points).



Name: Channel_1.Device_1

Update Rate (ms.): 1000

Time Bias (min.): 0

Percent Deadband: 0

Language ID: 1033

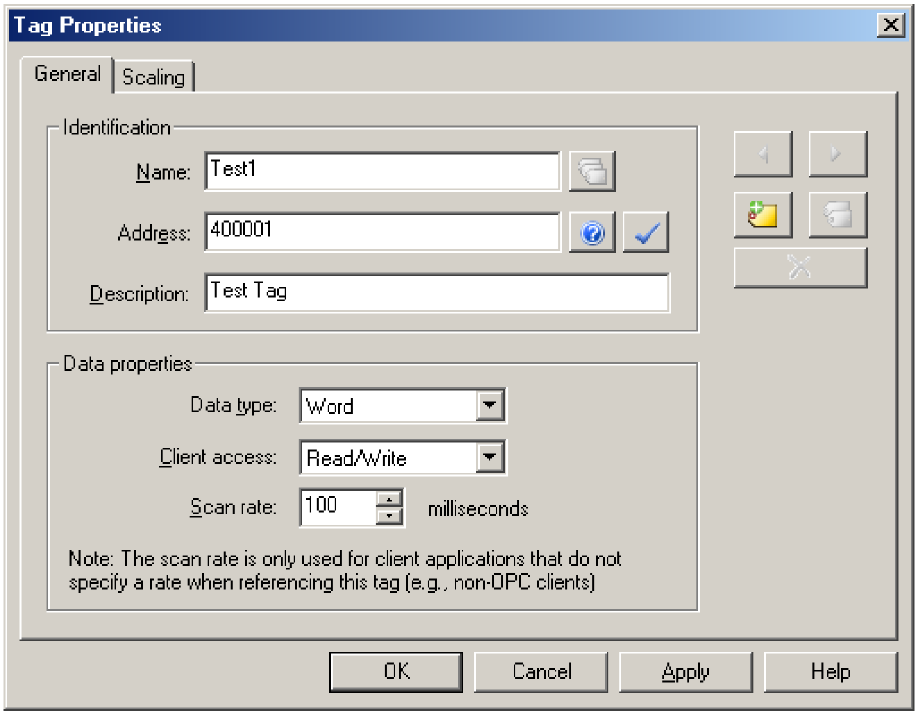
Update Notification: OPC 2.0

☑ Active State

Keep Alive Rate (ms): 0

**Figure 13: OPC Client General Group Properties**

For non-OPC Clients, a scan rate will be associated with each tag. This scan rate can only be changed after the client has released the item, and will be applied the next time that the client requests it. Different scan rates can be assigned to each item to allow for disparities between the polling requirements of critical and static data.



Name: Test1

Address: 400001

Description: Test Tag

Data type: Word

Client access: Read/Write

Scan rate: 100 milliseconds

Note: The scan rate is only used for client applications that do not specify a rate when referencing this tag (e.g., non-OPC clients)

**Figure 14: Scan Rate on a Tag for a Non-OPC Client Connection**

**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 server 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 of 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.

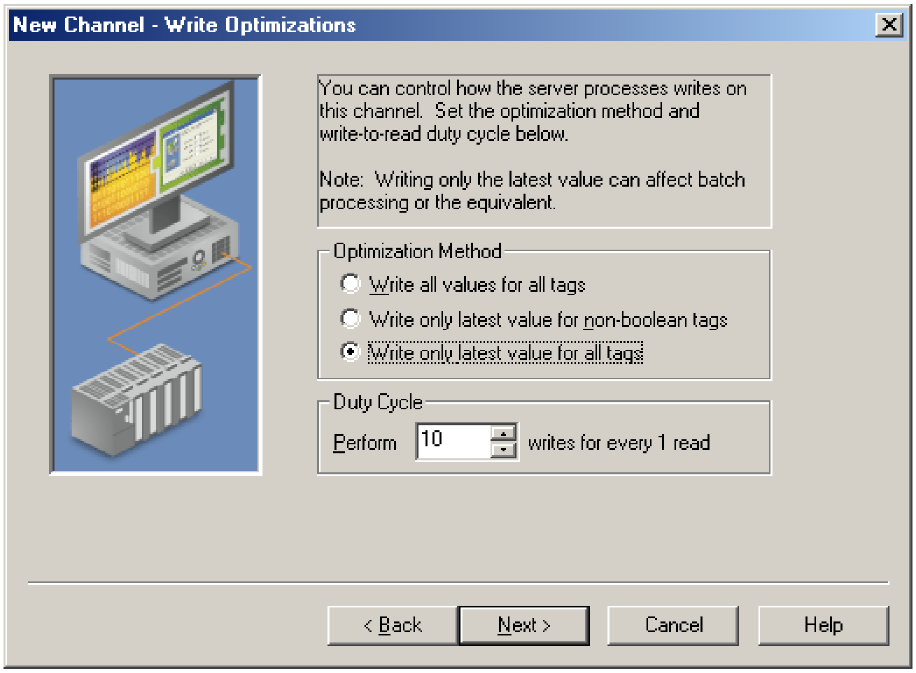
## 3.4 Managing Writes

When a client writes data to a device, the server will stop polling the device in order 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.



New Channel - Write Optimizations

You can control how the server processes writes on this channel. Set the optimization method and write-to-read duty cycle below.

Note: Writing only the latest value can affect batch processing or the equivalent.

Optimization Method

○ Write all values for all tags

○ Write only latest value for non-boolean tags

● Write only latest value for all tags

Duty Cycle

Perform 10 writes for every 1 read

**Figure 15: Write Optimizations**

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 all values for all tags."

The Duty Cycle specifies how many writes will be done 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 for efficient communication. Users must keep these factors in mind in order to apply the best optimization technique and plan for the best performance.