D

Appendix D
Using the H2 Series EBC with KEP*Direct* OPC Server

In This Appendix. . . .

Introduction to KEP*Direct* OPC Server

Introduction to OPC

OPC, OLE (**O**bject **L**inking and **E**mbedding) for Process Control, is an industry standard created by a number of worldwide leading hardware and software suppliers in cooperation with Microsoft. The OPC Data Access specification, as maintained by the OPC Foundation, is a non–proprietary technical specification that defines a set of standard interfaces based upon Microsoft's OLE/COM technology. An OPC server (driver) allows items such as distributed control systems, programmable logic controllers, I/O systems and smart field devices to communicate with a wide range of HMI/SCADA (client) software packages residing on a PC. Traditionally, each software or application developer was required to write a custom interface, or server/driver, to exchange information with hardware field devices. OPC eliminates this requirement allowing manufacturing customers true plug and play connectivity and the freedom to choose products based on their automation requirements.

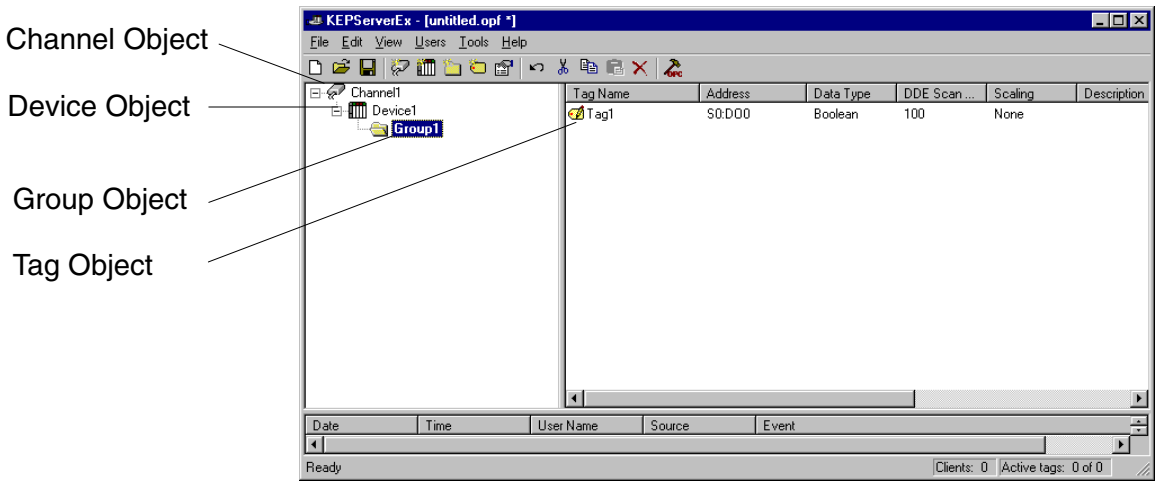
DDE Support

While KEP***Direct*** is first and foremost an OPC server, KEP***Direct*** recognized that a number of legacy applications still depend upon DDE for their underlying client server technology. Early in the development of Windows, Microsoft provided a generic client server technology called DDE (Dynamic Data Exchange). DDE did provide a basic architecture that would allow many windows applications from a wide range of vendors to share data. But there was one problem, DDE was not designed for the industrial market lacking much of the speed and robustness desired in an industrial setting. However, this did not stop DDE from becoming a dominant client/server architecture, largely due to its availability in most windows applications.

KEP*Direct*

KEP***Direct*** Enhanced OPC/DDE Server is a 32 bit windows application that provides a means of bringing data and information from a wide range of industrial devices and systems into client applications on your Windows PC. KEP***Direct*** falls under the category of a "Server" application. It is very common to hear the term "client/server application" in use across many software disciplines and business segments. In the industrial market, it has usually come to mean the sharing of manufacturing or production data between a variety of applications ranging from human machine interface software and data historians, to large MES and ERP applications.

At a high level, the KEP***Direct*** OPC Server is comprised of several objects that are described on the next page.



Channel Object: Each protocol or driver used in a KEP*Direct* project is referred to as a channel. A channel refers to a specific communications driver. A KEP*Direct* project can consist of many channels each with unique communications drivers or each with the same communications driver.
Each channel name must be unique in a KEP*Direct* application. The channel name entered here will be part of the OPC browser information.

Device Object: Unlike the channel name, ”Device names” can be the same from one channel to the next. The device name is a user defined logical name for the device. The device name and channel name will be part of the OPC browser information as well as a DDE item name. Within an OPC client the combination of channel name and device name would appear ”ChannelName.DeviceName”.

Group Object: KEP*Direct* allows tag groups to be added to your project. Tag groups allow you to tailor the layout of OPC data in logical groupings that fit the needs of your application. Using tag groups allows multiple sets of identical tags to be added under the same device. This can be very convenient when a single device handles a number of similar machine segments. From an OPC client standpoint, the use of tag grouping allows you to segregate your OPC data into smaller tag lists, which can make finding a specific tag easier when browsing the server.

Tag Object: KEP*Direct* allows both dynamic tags, (tag entered directly at the OPC client that specify device data) and user defined tags. User defined tags have the benefit of allowing the tag to be browsed from an OPC client that supports tag browsing. User defined tags also support tag scaling. Unlike many of the dialogs you will find in KEP*Direct*, the tag properties dialog has a number of features that are driven by icons. The tag name is part of the OPC browse data. Tag names must be unique within a given device branch or tag group branch. If your application is best suited by using blocks of tags with the same names, use tag groups to segregate the tags.

KEP*Direct* Project: Adding and Configuring a Channel

Running the Server

KEP***Direct***, like any OPC server, can be started a number of ways. One of the benefits of OPC technology is that your OPC client can automatically invoke the server when it attempts to connect and collect data from it. In order for this automatic mode of operation to occur you must first create and configure a project. Once you have created a project, KEP***Direct*** will automatically select the most recently used project when it is invoked by an OPC client.

Initially however, you need to manually invoke KEP***Direct*** using either the desktop icon, if you chose to install it, or by selecting KEP***Direct*** from the windows start menu. Depending on any changes you may have made to the appearance of KEP***Direct***, once invoked you should be presented with the following interface. To learn more about the various elements of the user interface see (Basic KEP***Direct*** Components).

While discussing how to start KEP***Direct*** its important to understand what the system requirements are for running the server. KEP***Direct*** has been designed to place as little strain on your system as possible.

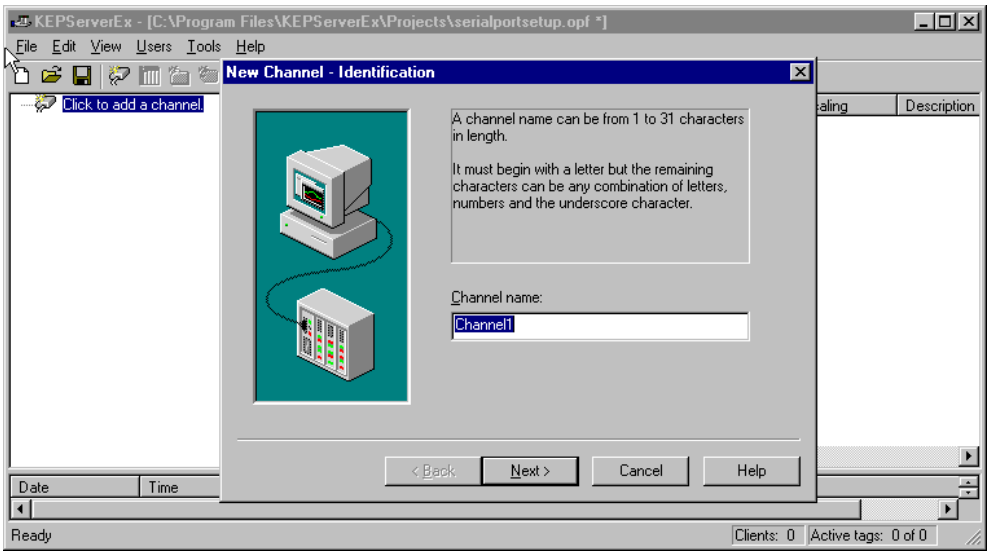
Recommended System Requirements:

400Mhz Pentium
64 Megs of Ram
10 Megs of Hard Disk Space
Windows NT(SP6a)/2000 (Strongly recommended for industrial settings)
Available Ethernet Card

Adding a Channel

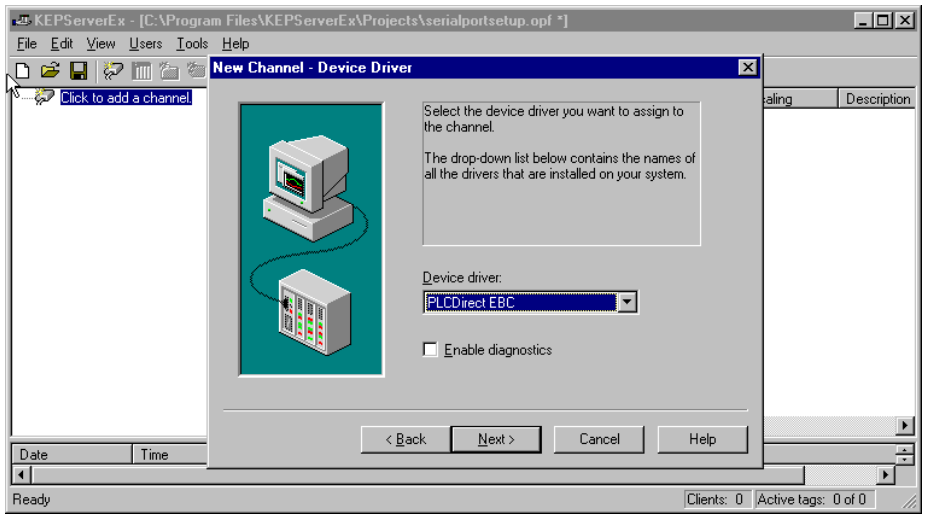
A channel refers to a specific communications driver. A KEP***Direct*** project can consist of many channels each with unique communications drivers or each with the same communications driver. Depending on the driver or drivers you have installed you can define a number of channels within a single project. A channel acts as the basic building block of an OPC link. Properties like communications port, baud rate, and parity are contained at the channel level. Each channel name must be unique in a KEP***Direct*** project. The channel name can be up to 31 characters long.

To add a new channel to your project you can use the Edit menu > New Channel, the Toolbar Add Channel, or the “Click to add a channel” dialog.



Selecting the Device Driver

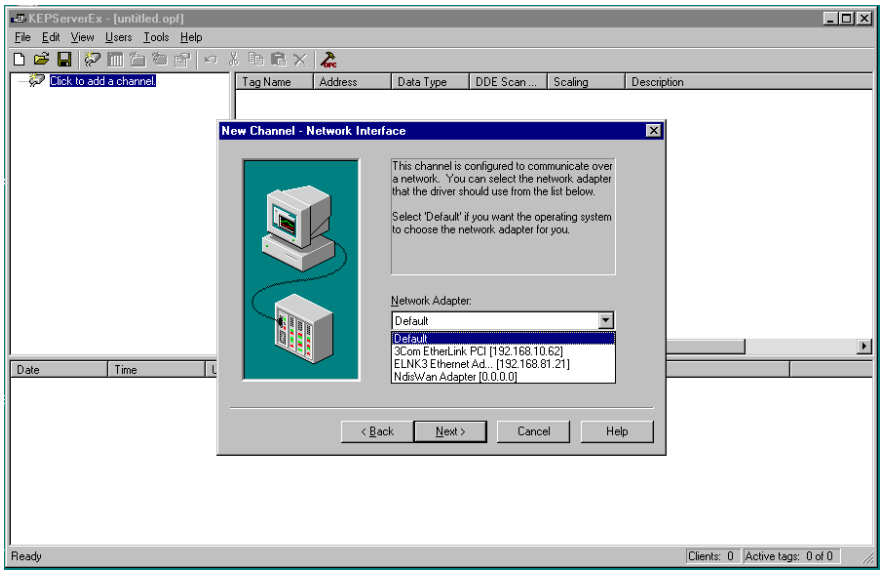
Select the device driver you want to assign to the channel. A driver list will be presented displaying all of the device drivers that are installed in your system.



Selecting the ”Enable diagnostics” check box will enable diagnostic information to be available to your OPC application for this channel. With diagnostic functions enabled, diagnostic tags are available for use within client applications. In addition to diagnostic tags, a diagnostic window is also available when this feature is enabled. The diagnostic features of KEP*Direct* do require a minimal amount of overhead processing. For this reason it is recommended that you only use the diagnostic features when needed and disable them when not in use which is the default case.

Selecting the Network Adapter

The Network Interface selection allows you to select a specific NIC card for the AutomationDirect EBC Ethernet driver to use based on the NIC name or its assigned IP address. By selecting a specific NIC interface you will be able to force the driver to send all Ethernet communication through the specified NIC. If you do not know which NIC you should use, select the ”Default” condition.



Setting the Server Writes Optimizations

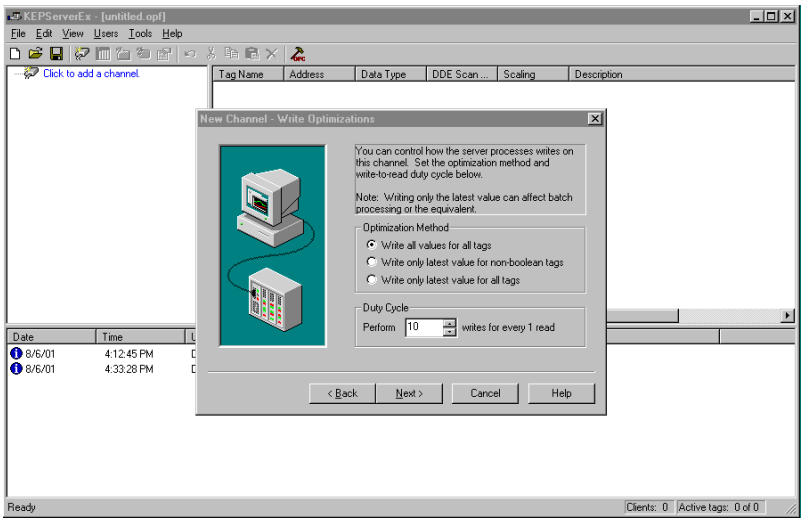
As with any OPC server, writing data to your device may be the most important aspect of your application. Insuring that the data written from your OPC client application gets to the device in a timely manners is the goal of the server. KEP*Direct* provides a number of optimization settings that can be used to tailor the server to meet the needs, and improve the responsiveness of your application.

There are currently three write optimization modes. The following is a brief description of the modes. For a detailed explanation, refer to the “Channel Properties – Write Optimizations” section in the KEP*Direct* on–line help file.

NOTE: We strongly suggest that you characterize your application for compatibility with these write optimization enhancements before using them in a production environment.

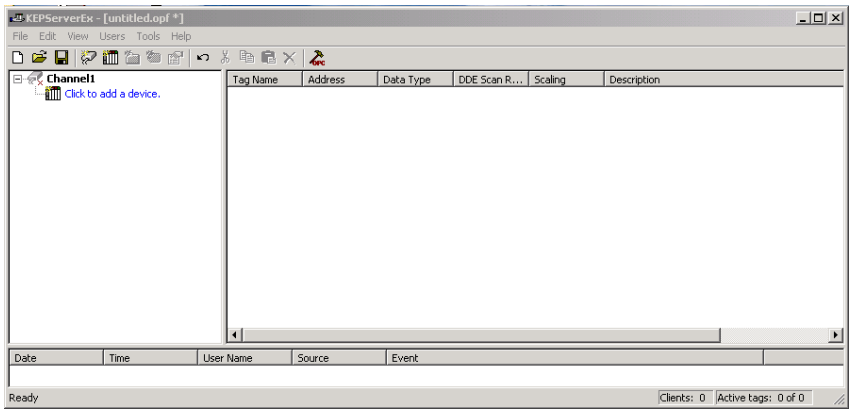
The default mode, ”**Write all values for all tags**” will force the server to attempt to write every value to the controller. This mode insures that everything written from your OPC client applications will be sent to the target device. While writing every value to the device may seem like the best course of action, there are a number of applications where writing every value, many of which may be the same value, over and over may be simply a waste of communications bandwidth.
The ”**Write only latest value for non–boolean tags**” allows any value that is not a boolean value to be updated in the server’s internal write queue and will then be sent to the device at the next possible opportunity. This can dramatically improve the overall performance of your application. This feature must be used with a clear understanding of how it will affect the operation of your application.
The final write optimization mode, ”**Write only the latest value for all tags**”, takes the operation described for the second mode and applies it to all tags.

The **Duty Cycle** selection allows you to control the ratio of write operations to read operations. By default the duty cycle is set to ten. This means that ten writes will occur for each read operation. If your application is doing a large number of continuous writes but you need to insure that read data is still given time to process, you may want to reduce the Duty Cycle. A setting of one will result in one read operation for every write operation. In all cases if there are no write operations to perform, reads will be processed continuously.



Saving the New Channel Settings

With **Channel1** added to the server, the KEP*Direct* window will appear as follows:



Note that the channel is shown using the channel name given, but it also has a small red ”x” below the channel icon. The red ”x” indicates that the channel does not contain a valid configuration. **Channel1** is not valid because a device has not yet been added to the channel.

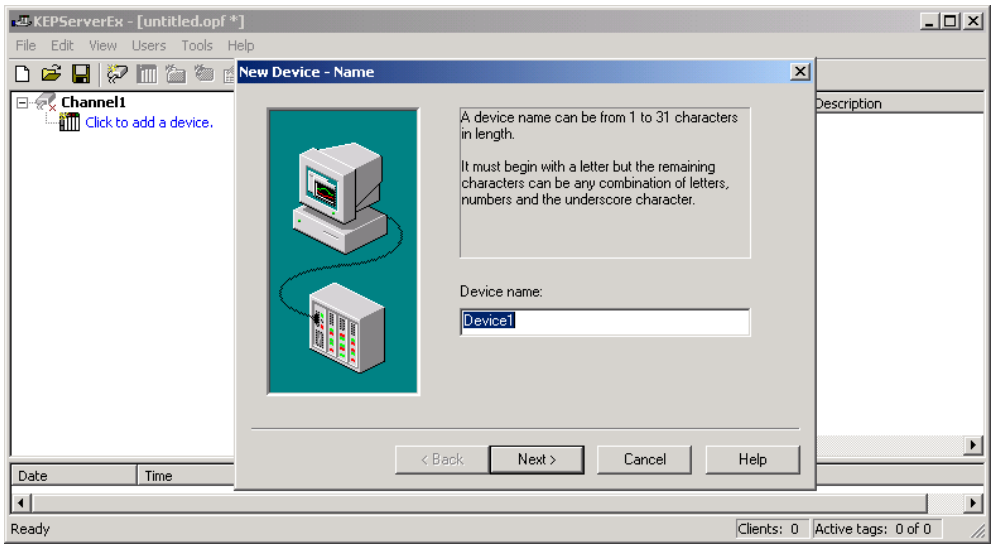
Using Multiple Channels in a Project

KEP*Direct* supports the use of multiple channels. As you add channels to your project you can specify either the same communications driver or different communications drivers. Most communication drivers offered by KEP*Direct* support operation on up to 16 communications ports or ethernet network connections simultaneously. By defining multiple channels you can improve the overall performance of you application. In the case of either a serial driver or Ethernet driver using multiple channels allows you to spread large communications loads across the multiple channels. A good example of this would be a serial driver that is being used to communicate with eight devices on the serial line. Normally the communications driver used in this application would be responsible for gathering data from all eight devices in a round robin fashion. If this same application is reconfigured to use multiple channels assigned to multiple communications ports, the device load can be divided across the channels. The end result is reduce work load on each channel and dramatic improvements in the responsiveness of your application. The need to use multiple channels is dependent solely on the needs of your application. In either case there is no additional cost involved to use a licensed driver on multiple communications or Ethernet ports.

KEP*Direct* Project: Adding and Configuring a Device

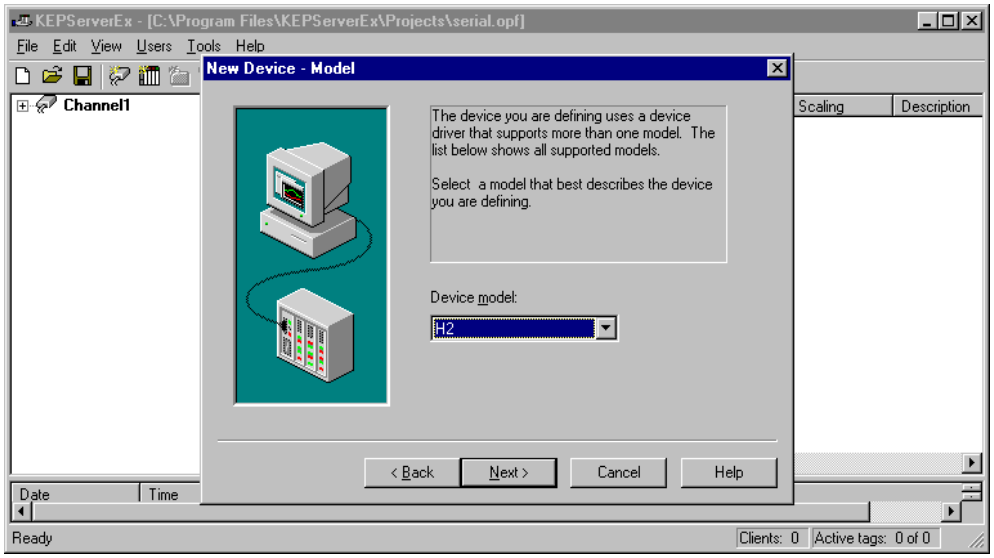
Adding a Device

Once a channel has been configured in a KEP***Direct*** project, a device must be added to the channel. Devices represent PLCs, I/O devices or other hardware that the server will communicate with. Device selection is restricted by the device driver the channel is using.
To add a device to a channel, select the desired channel and use the Edit menu > New Device, the Toolbar Add Device, or the “Click to add a device” dialog.



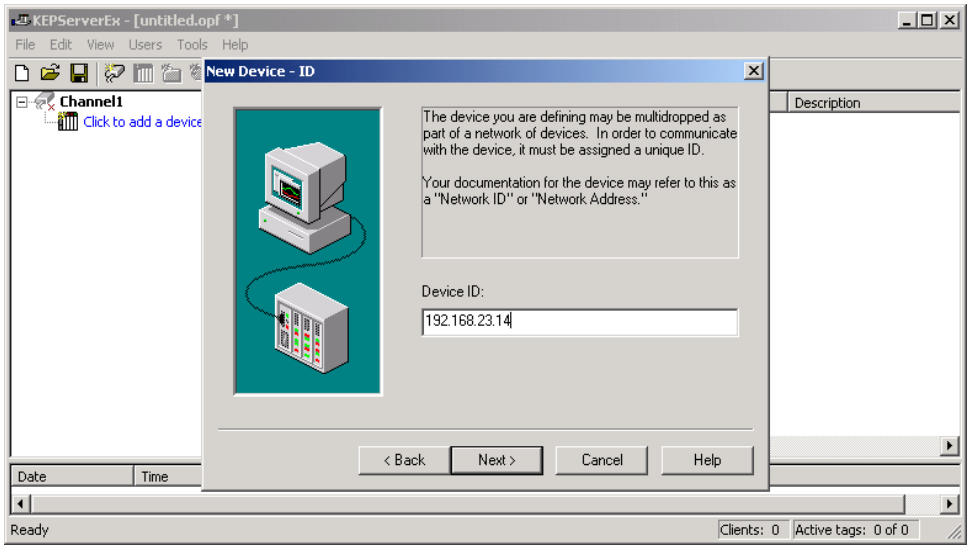
Selecting the Device Model

The ”Model” parameter allows you to select the specific type of the device associated with a device ID. The contents of the model selection drop down will vary depending on the chosen communication driver.



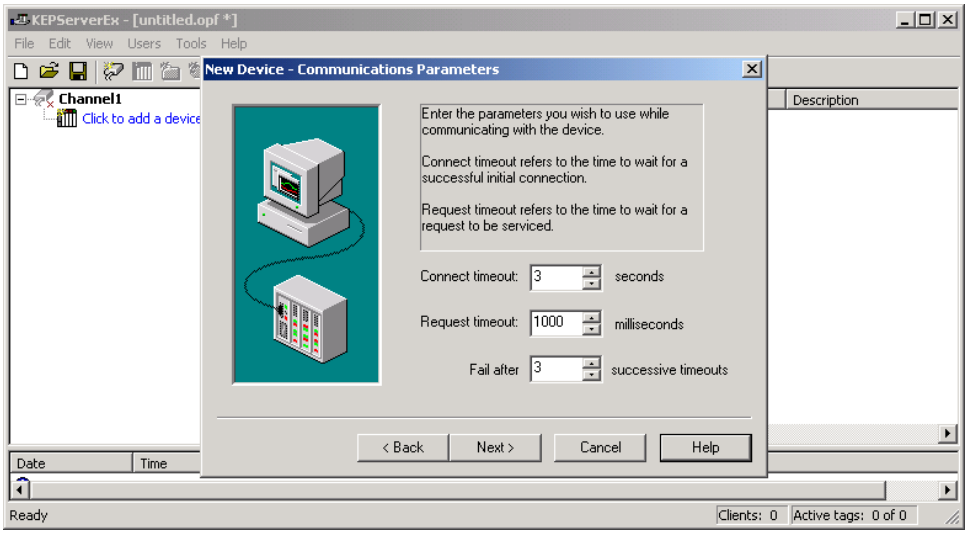
Selecting the Device Model

The ”Device ID” parameter allows you to specify the driver specific station or node address for a given device. Since the Automationdirect EBC driver is an Ethernet based driver, a unique and valid TCP/IP address must be entered. IPX protocol is not supported.



Setting the Device Timeout Properties

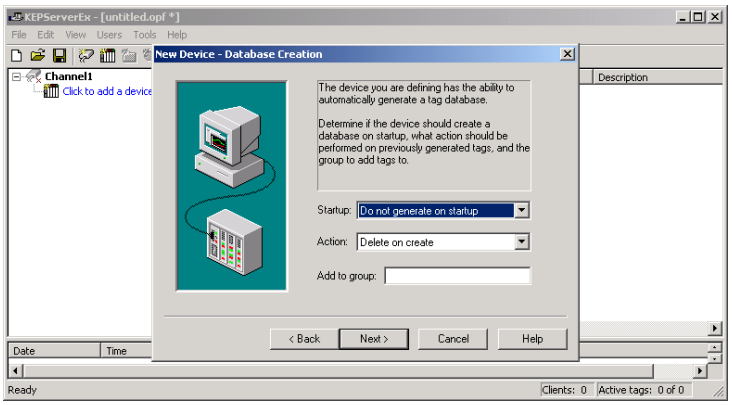
Device timeout parameters allow a driver’s response to error conditions to be tailored to the needs of your application. The timeout parameters are specific to each device you configure. Each of the field parameters is defined in detail in the “Device Properties – Timeout” section in the KEP*Direct* on–line help file.



The **Connection timeout**: allows the time required to establish a socket connection to a remote device to be adjusted. The **Request timeout**: is used by all drivers to determine how long the driver will wait for a response from the target device. The **Fail after** parameter is used to determine how many times the driver will retry a communications request before considering the request to have failed. If your environment is prone to noise induced communications failures you may want to increase the number of **retries** the driver performs.

Automatic OPC Tag Database Generation

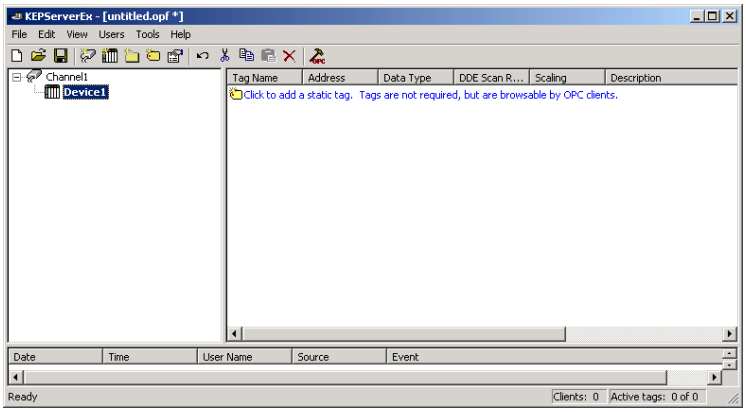
The automatic OPC tag database generation features of KEP*Direct* have been designed to make the setup of your OPC application a Plug and Play operation. Since the Automationdirect EBC communication driver supports this feature, you can configure it automatically build a list of OPC tags within KEP*Direct* that correspond to device specific data. The automatically generated OPC tags are then browsable from your OPC client. The OPC tags that are generated are dependent upon the nature of the supporting driver. Each field selection is defined in detail in the "Automated OPC Tag Base Generation" section in the KEP*Direct* on–line help file.



The "**Automatic tag database generation on device startup**" selection allows you to configure when OPC tags will be automatically generated. There are three possible selections. The default condition, "**Do not generate on startup**", will prevent the driver from adding any OPC tags to tag space of KEP*Direct*. The selection "**Always generate on startup**", will cause the driver to always evaluate the device for tag information and to add OPC tags to the tag space of the server each time the server is launched. The final selection "**Generate on first startup**" will cause the driver to evaluate the target device for tag information the first time this KEP*Direct* project is run and to add any OPC tags to the server tag space as needed. When the automatic generation of OPC tags is selected, any tags that are added to the server's tag space must be saved with the project. You can configure your KEP*Direct* project to auto save from the Tools > Options menu.

Saving the New Device Settings

With **Device1** added to **Channel1**, the KEP*Direct* window will appear as follows:



KEP*Direct* Project: Adding Tags to the Project

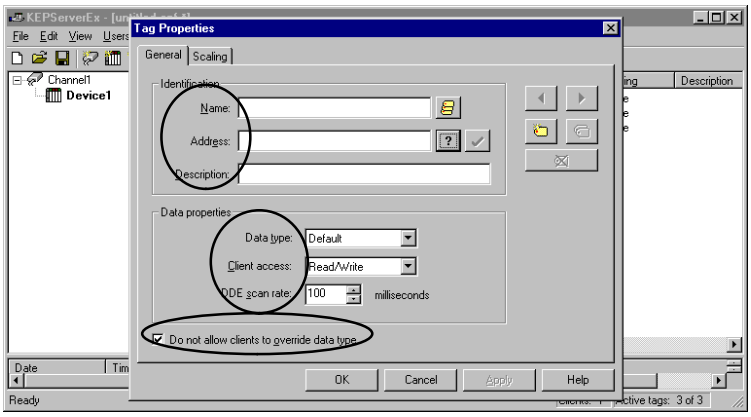
There are two ways to get data from a device to your client application using KEP***Direct***. The first method, and most common method, of defining tags is called **User Defined Tags**. This requires that you define a set of tags in the server project and then use the name you assigned to each tag as the item of each OPC/DDE link between the client and the server. The primary benefit to this method is that all user defined tags are available for browsing within OPC clients. Additionally, user defined tags also support scaling.

The second method of defining tags is called **Dynamic Tags**. Dynamic tags allow you to define tags in the client application. Instead of providing the server with a tag name as the OPC/DDE item, you would provide the device address (and optionally a data type). The server will create a tag for that location and start scanning for data automatically. KEP***Direct*** allows tag groups to be added to your project.

Tag groups allow you to tailor the layout of OPC data in logical groupings that will fit the needs of your application. Using tag groups allows multiple sets of identical tags to be added under the same device. This can be very convenient when a single device handles a number of similar machine segments. From an OPC client standpoint, the use of tag grouping allows you to segregate your OPC data into smaller tag lists, which can make finding a specific tag easier when browsing the server.

User Defined Tags

Each field selection is defined in detail in the **Tag Properties** section in the KEP***Direct*** on–line help file. A brief description of each is listed below.



The tag **Name:** parameter allows you to enter the string that will represent the data available from this tag. The tag name can be up to 31 characters in length. While using long descriptive names is generally a good idea, keep in mind that some OPC client applications may have a limited display window when browsing the tag space of an OPC server. The tag name is part of the OPC browse data. Tag names must be unique within a given device branch or a tag group branch. If your application is best suited by using blocks of tags with the same names, use tag groups to segregate the tags.

The **Address:** parameter allows you to enter the desired driver address for this tag. To determine how an address should be entered, you can use the **Hints button** next to the address parameter. Hints provide a quick reference guide to the address format of the driver. Once you have entered an address you can test it by using the check address button. When pressed, the check address button attempts to validate the address with the driver. If the driver accepts the address as entered no message will be displayed. If an error is detected a pop–up will inform you of the error. Keep in mind that some errors will be related to the data type selection and not the address string.

The **Description:** parameter allows you to attach a comment to this tag. A string of up to 64 characters can be entered for the description. If you are using an OPC client that supports Data Access 2.0 Tag Properties, the description parameter will be accessible from the Item Description property of the tag.

The **Data Type:** selection allows you to specify the format of the tag's data as it is found in the physical device. The data type setting is an important part of how a communication driver reads and writes data to a device. For many drivers the data type of a particular piece of data is rigidly fixed.
The available data type selections are:

- **Default** – This type allows the driver to choose its default data type see the specific driver help for details
- **Boolean** – Single bit data On or Off
- **Char** – Signed 8 bit data
- **Byte** – Unsigned 8 bit data
- **Short** – Signed 16 bit data
- **Word** – Unsigned 16 bit data
- **Long** – Signed 32 bit data
- **Dword** – Unsigned 32 bit data
- **Float** – 32 bit Real value IEEE format
- **String** – Null terminated ASCII string
- **Double** – 64 bit Real value IEEE format
- **BCD** – Two byte packed BCD value range is 0 – 9999
- **LBCD** – Four byte packed BCD value range is 0 – 99999999

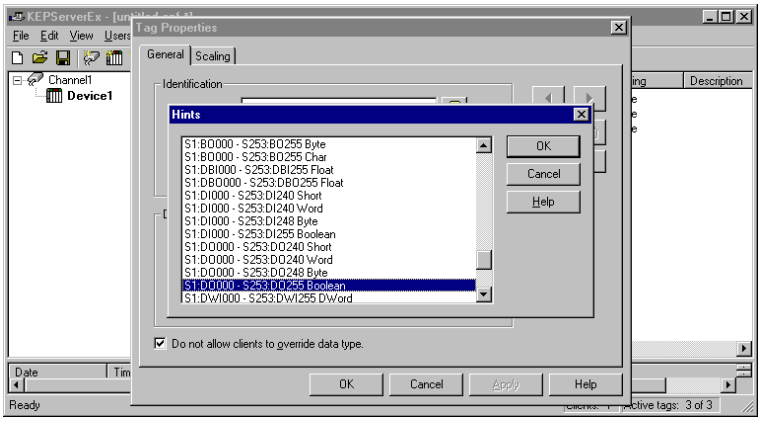
The **Client access:** selection allows you to specify whether this tag is Read Only or Read/Write. By selecting Read Only you can prevent client applications from changing the data contained in this tag. By selecting Read/Write you are allowing client applications to change this tag's value as needed.

The **DDE scan rate:** parameter allows you to specify the the update interval for this tag when used in a DDE client. OPC clients can control the rate at which data is scanned by using the update rate that is part of all OPC groups.

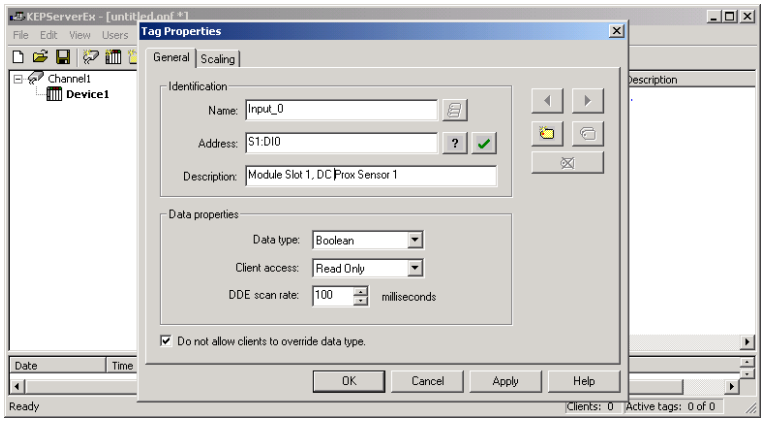
The **Do not allow client to override data type** selection allows you force OPC clients to use the data type you have specified for this tag. OPC clients can specify how they desire to view the data from a particular tag.

Creating a User Define Tag

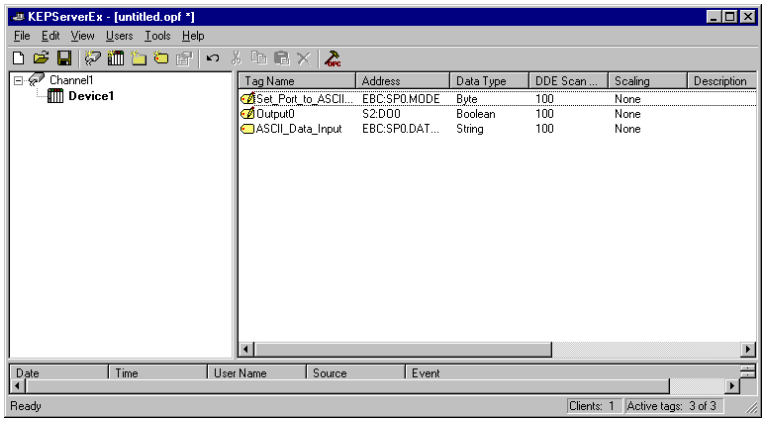
To determine how an address should be entered, use the Hints button "**?**" to the right of the address field. Hints provide a quick reference guide to the address format of the driver.



Once you have entered an address you can test it using the check address "✓" button. When pressed, the check address button attempts to validate the address with the driver. If the driver accepts the address as entered, no message will be displayed. If an error is detected, a pop–up window will inform you of the error. Keep in mind that some errors will be related to the data type selection and not the address string. Below is an example of a valid tag properties.



The window below shows a valid configured channel, device and several user defined tags.



H2 Series EBC I/O Addressing

I/O slots must be individually addressed in the following form: S<ss>:<t><nn> where ss is the slot number (0 to 8), t is the address type (X, Y, K, V, DI, D0, WI, W0, etc.), and nn is the address. The address ranges from 0 to an upper limit determined by the module occupying the slot.

I/O Type	Syntax	Data Type
Discrete Inputs	X or DI<nn> nn = Bit Number (decimal)	**Boolean**, Byte, Char, Word, Short, DWord, Long
Discrete Outputs	Y or DO<nn> nn = Bit Number (decimal)	**Boolean**, Byte, Char, Word, Short, DWord, Long
Byte Inputs	BI<nn> nn = Bit Number (decimal)	**Byte**, Char
Byte Outputs	BO<nn> nn = Bit Number (decimal)	**Byte**, Char
Word Inputs	K or WI<nn> nn = Bit Number (decimal)	**Word**, Short
Word Outputs	V or WO<nn> nn = Bit Number (decimal)	**Word**, Short
DWord Inputs	DWI<nn> nn = Bit Number (decimal)	**DWord**, Long
DWord Outputs	DWO<nn> nn = Bit Number (decimal)	**DWord**, Long
Float Inputs	FI<nn> nn = Bit Number (decimal)	**Float**
Float Outputs	FO<nn> nn = Bit Number (decimal)	**Float**
Double Inputs	DBI<nn> nn = Bit Number (decimal)	**Float**
Double Outputs	DBO<nn> nn = Bit Number (decimal)	**Float**

H2–EBC I/O Addressing Example

Each field selection is defined in detail in the “Tag Properties” section in the KEP*Direct* on–line help file.

H2 Series EBC Module	Slot 0 8 Inputs Addresses S0:X0 to S0:X7	Slot 1 32 Inputs Addresses S1:X0 to S1:X31	Slot 2 4 Analog Inputs Addresses S2:K0 to S2:K3	Slot 3 8 Outputs Addresses S3:Y0 to S3:Y7	Slot 4 16 Outputs Addresses S4:Y0 to S4:YI5	Slot 5 8 Analog Outputs Addresses S5:V0 to S5:V7