# Dataforth isoLynx Driver

# Table of Contents

# Dataforth isoLynx Driver

Help version 1.022

## CONTENTS



## Overview

The Dataforth isoLynx Driver provides a reliable way to connect Dataforth isoLynx devices to OPC client applications; including HMI, SCADA, Historian, MES, ERP, and countless custom applications. It is intended for use with all ISOLYNX SLX100 data acquisition systems.

This driver was created in partnership with DATAFORTH and DATAFORTH endorses the use of this driver as the ISOLYNX SLX100 official OPC interface for their products. In addition, this driver uses DATAFORTH supplied communication software modules that encapsulates the low-level communication details of the isoLynx Command Protocol.

## Setup

This driver uses the ISOLYNX Data Acquisition Library, which is comprised of the following dynamic link libraries.

- LYNXW32.DLL
- _ISOLYNX.DLL
- _SUPER.DLL
- SUPERCOM.DLL
- SCRS232.DLL
- SCTCPIP.DLL.

When the driver is installed, the server installs these components into the Windows system directory.

### Supported Devices

All ISOLYNX SLX100 data acquisition systems that support the ISOLYNX protocol.

### Supported Protocols
ISOLYNX protocol over serial lines and Ethernet.

### Networking
This driver supports communications over serial lines and Ethernet. *For more information on ISOLYNX communications and connections, refer to Section 7.0 of ISOLYNX hardware user's manual.*

### Channel and Device Limits
The maximum number of channels supported by this driver is 100. The maximum number of devices supported by this driver is 16 per channel.

### Device Configuration
Each device on a network must be configured with a unique Address/ID. *For information on setting each device's network address, refer to "Network Address Selection" in the ISOLYNX hardware user's manual.*

Each additional panel within the system must also have its individual address configured. For more information, refer to ISOLYNX hardware user's manual section on "Analog I/O expansion panel-Address selection" and "Digital I/O expansion panel-Network Address Selection".

The final step in the device setup process is to use the Dataforth supplied configuration utility to configure both the Interface and the I/O. For more information, refer to section 3.5 "Sample Applications" in the software user's manual. It is possible to configure an I/O channel incorrectly or to configure a channel with no I/O module physically present. The best way to verify that the I/O configuration is correct is to perform an auto create of the tag database and then match the tags generated to the actual hardware I/O.

The Ethernet interface board supports up to four simultaneous connections at a time. This means that there can only be up to four devices with same IP address running in the server at one time. These devices can be assigned to one or more channels.

When configuring a device's IP address and associated items using Dataforth's configuration utility, pay attention to the Keep Alive item. This item defines how long (in seconds) the Ethernet interface board keeps a connection alive when no activity is seen. When using multiple connections to the same device, it is strongly recommended that this item be set to a value of 1. If not, the server may not be able to re-establish a connection to the device when a break occurs in the connection.

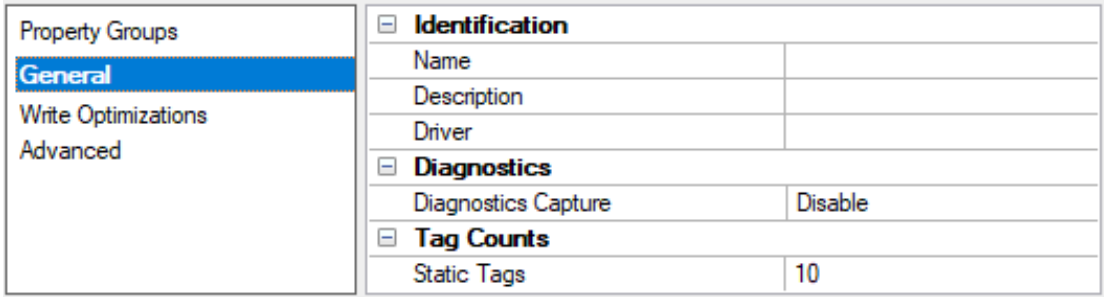
**Notes**:

1. This driver provides multi-threaded processing for optimum performance.
2. Each physical device to be polled must be represented by a device object in the server.

**Important:** The communication properties selected must match those set up with the Dataforth configuration utility. *For more information, refer to 3.5 "Sample Applications" and "Configuration Sample" in the software manual.*

## Channel Properties — General
This server supports the use of multiple simultaneous communications drivers. Each protocol or driver used in a server project is called a channel. A server project may consist of many channels with the same communications driver or with unique communications drivers. A channel acts as the basic building block of an

OPC link. This group is used to specify general channel properties, such as the identification attributes and operating mode.




## Identification

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

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

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

## Diagnostics

**Diagnostics Capture**: When enabled, this option makes the channel's diagnostic information available to OPC applications. Because the server's diagnostic features require a minimal amount of overhead processing, it is recommended that they be utilized when needed and disabled when not. The default is disabled.
**Note:** This property is not available if the driver does not support diagnostics.
*For more information, refer to Communication Diagnostics in the server help.*

## Diagnostics

**Diagnostics Capture**: When enabled, this option allows the usage of statistics tags that provide feedback to client applications regarding the operation of the channel. Because the server's diagnostic features require a minimal amount of overhead processing, it is recommended that they be utilized when needed and disabled when not. The default is disabled.
**Note:** This property is not available if the driver does not support diagnostics.

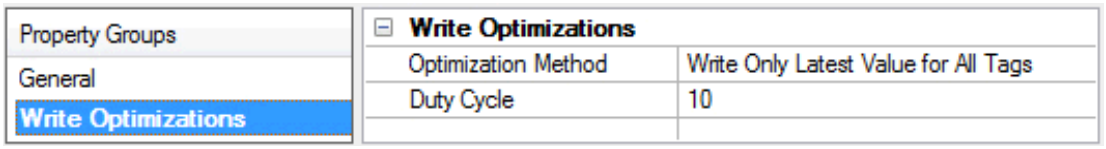
*For more information, refer to Statistics Tags in the server help.*

### Tag Counts

**Static Tags**: Provides the total number of defined static tags at this level (device or channel). This information can be helpful in troubleshooting and load balancing.

## Channel Properties — Write Optimizations

The server must ensure that the data written from the client application gets to the device on time. Given this goal, the server provides optimization properties to meet specific needs or improve application responsiveness.




### Write Optimizations

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

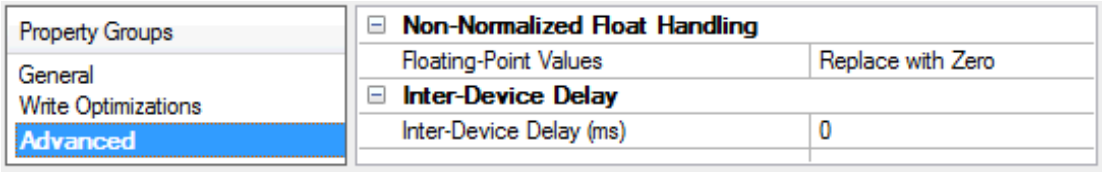
- **Write All Values for All Tags**: This option forces the server to attempt to write every value to the controller. In this mode, the server continues to gather write requests and add them to the server's internal write queue. The server processes the write queue and attempts to empty it by writing data to the device as quickly as possible. This mode ensures that everything written from the client applications is sent to the target device. This mode should be selected if the write operation order or the write item's content must uniquely be seen at the target device.
- **Write Only Latest Value for Non-Boolean Tags**: Many consecutive writes to the same value can accumulate in the write queue due to the time required to actually send the data to the device. If the server updates a write value that has already been placed in the write queue, far fewer writes are needed to reach the same final output value. In this way, no extra writes accumulate in the server's queue. When the user stops moving the slide switch, the value in the device is at the correct value at virtually the same time. As the mode states, any value that is not a Boolean value is updated in the server's internal write queue and sent to the device at the next possible opportunity. This can greatly improve the application performance.
  **Note**: This option does not attempt to optimize writes to Boolean values. It allows users to optimize the operation of HMI data without causing problems with Boolean operations, such as a momentary push button.
- **Write Only Latest Value for All Tags**: This option takes the theory behind the second optimization mode and applies it to all tags. It is especially useful if the application only needs to send the latest value to the device. This mode optimizes all writes by updating the tags currently in the write queue before they are sent. This is the default mode.

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

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

## Channel Properties — Advanced

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




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

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

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

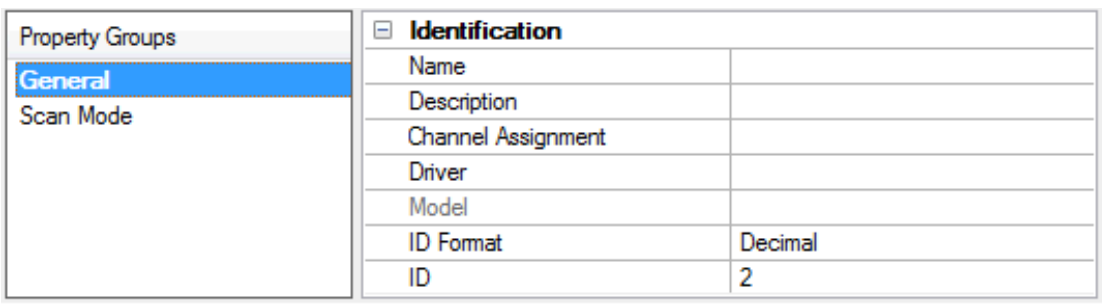
*For more information on the floating-point values, refer to "How To ... Work with Non-Normalized Floating-Point Values" in the server help.*

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

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

## Device Properties — General

A device represents a single target on a communications channel. If the driver supports multiple controllers, users must enter a device ID for each controller.




### Identification

**Name**: Specify the name of the device. It is a logical user-defined name that can be up to 256 characters long and may be used on multiple channels.

**Note**: Although descriptive names are generally a good idea, some OPC client applications may have a limited display window when browsing the OPC server's tag space. The device name and channel name become part of the browse tree information as well. Within an OPC client, the combination of channel name and device name would appear as "ChannelName.DeviceName".
*For more information, refer to "How To... Properly Name a Channel, Device, Tag, and Tag Group" in server help.*

**Description**: Specify the user-defined information about this device.
Many of these properties, including Description, have an associated system tag.

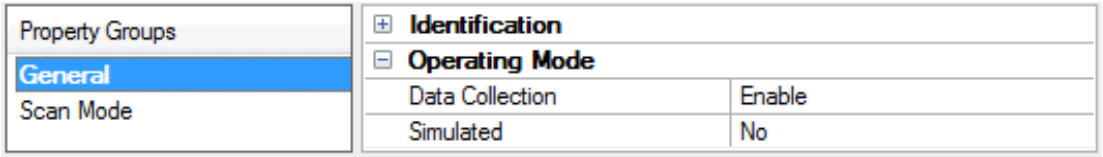
**Channel Assignment**: Specify the user-defined name of the channel to which this device currently belongs.

**Driver**: Selected protocol driver for this device.

**Model**: Specify the type of device that is associated with this ID. The contents of the drop-down menu depend on the type of communications driver being used. Models that are not supported by a driver are disabled. If the communications driver supports multiple device models, the model selection can only be changed when there are no client applications connected to the device.
**Note:** If the communication driver supports multiple models, users should try to match the model selection to the physical device. If the device is not represented in the drop-down menu, select a model that conforms closest to the target device. Some drivers support a model selection called "Open," which allows users to communicate without knowing the specific details of the target device. *For more information, refer to the driver documentation.*

**ID**: Specify the device's driver-specific station or node. The type of ID entered depends on the communications driver being used. For many communication drivers, the ID is a numeric value. Drivers that support a Numeric ID provide users with the option to enter a numeric value whose format can be changed to suit the needs of the application or the characteristics of the selected communications driver. The format is set by the driver by default. Options include Decimal, Octal, and Hexadecimal.
**Note**: If the driver is Ethernet-based or supports an unconventional station or node name, the device's TCP/IP address may be used as the device ID. TCP/IP addresses consist of four values that are separated by periods, with each value in the range of 0 to 255. Some device IDs are string based. There may be additional properties to configure within the ID field, depending on the driver.

## Operating Mode




**Data Collection**: This property controls the device's active state. Although device communications are enabled by default, this property can be used to disable a physical device. Communications are not attempted when a device is disabled. From a client standpoint, the data is marked as invalid and write operations are not accepted. This property can be changed at any time through this property or the device system tags.

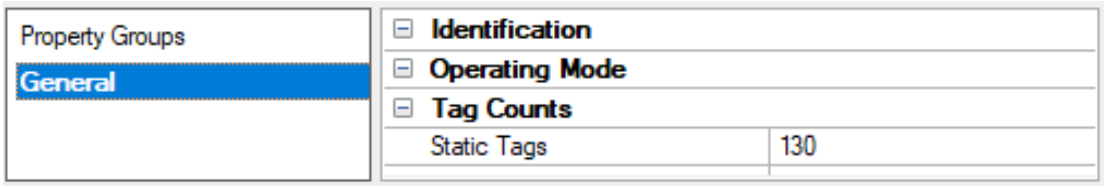
**Simulated**: Place the device into or out of Simulation Mode. In this mode, the driver does not attempt to communicate with the physical device, but the server continues to return valid OPC data. Simulated stops physical communications with the device, but allows OPC data to be returned to the OPC client as valid data. While in Simulation Mode, the server treats all device data as reflective: whatever is written to the simulated device is read back and each OPC item is treated individually. The data is not saved if the server removes the item (such as when the server is reinitialized). The default is No.

**Notes:**

1. Updates are not applied until clients disconnect and reconnect.
2. The System tag (_Simulated) is read only and cannot be written to for runtime protection. The System tag allows this property to be monitored from the client.
3. In Simulation mode, the item's memory map is based on client update rate(s) (Group Update Rate for OPC clients or Scan Rate for native and DDE interfaces). This means that two clients that reference the same item with different update rates return different data.
4. When a device is simulated, updates may not appear faster than one (1) second in the client.

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

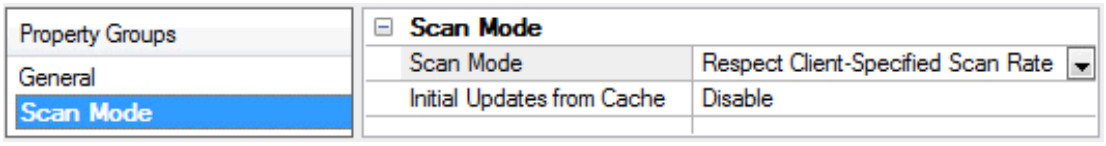
### Tag Counts




**Static Tags**: Provides the total number of defined static tags at this level (device or channel). This information can be helpful in troubleshooting and load balancing.

## Device Properties — Scan Mode

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




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

- **Respect Client-Specified Scan Rate**: This mode uses the scan rate requested by the client.
- **Request Data No Faster than Scan Rate**: This mode specifies the value set as the maximum scan rate. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
  **Note**: When the server has an active client and items for the device and the scan rate value is

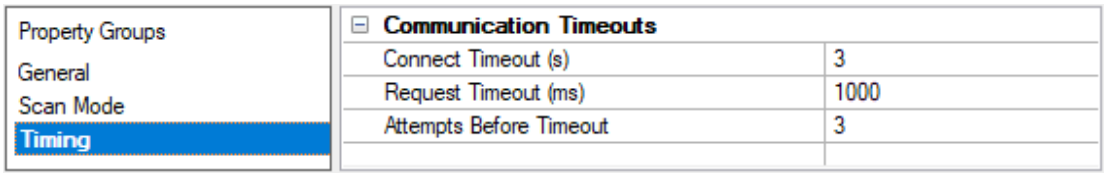
increased, the changes take effect immediately. When the scan rate value is decreased, the changes do not take effect until all client applications have been disconnected.

- **Request All Data at Scan Rate**: This mode forces tags to be scanned at the specified rate for subscribed clients. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
- **Do Not Scan, Demand Poll Only**: This mode does not periodically poll tags that belong to the device nor perform a read to get an item's initial value once it becomes active. It is the OPC client's responsibility to poll for updates, either by writing to the _DemandPoll tag or by issuing explicit device reads for individual items. *For more information, refer to "Device Demand Poll" in server help*.
- **Respect Tag-Specified Scan Rate**: This mode forces static tags to be scanned at the rate specified in their static configuration tag properties. Dynamic tags are scanned at the client-specified scan rate.

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

## Device Properties — Timing

The device Timing properties allow the driver's response to error conditions to be tailored to fit the application's needs. In many cases, the environment requires changes to these properties for optimum performance. Factors such as electrically generated noise, modem delays, and poor physical connections can influence how many errors or timeouts a communications driver encounters. Timing properties are specific to each configured device.




### Communications Timeouts

**Connect Timeout**: 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**: Specify 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 9999999 milliseconds (167 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.

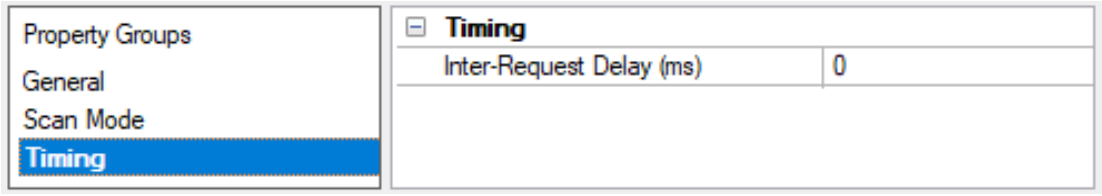
**Attempts Before Timeout**: Specify how many times the driver issues a communications request before considering the request to have failed and the device to be in error. The valid range is 1 to 10. The default is typically 3, but can vary depending on the driver's specific nature. The number of attempts configured for an

application depends largely on the communications environment. This property applies to both connection attempts and request attempts.

### Timing

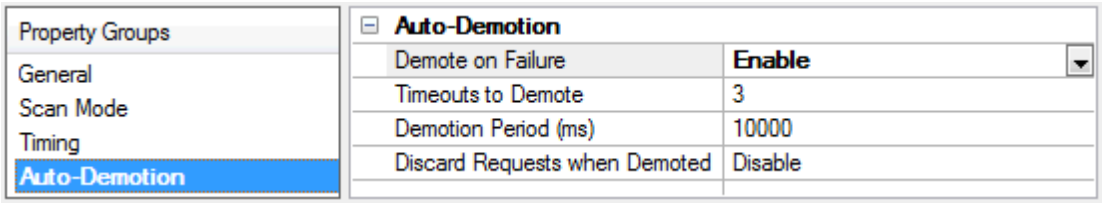
**Inter-Request Delay**: Specify how long the driver waits before sending the next request to the target device. It overrides the normal polling frequency of tags associated with the device, as well as one-time reads and writes. This delay can be useful when dealing with devices with slow turnaround 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.




## Device Properties — Auto-Demotion

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




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

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

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

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

**Discard Requests when Demoted**: Select whether or not write requests should be attempted during the off-scan period. Disable to always send write requests regardless of the demotion period. Enable to discard

writes; the server automatically fails any write request received from a client and does not post a message to the Event Log.

## Device Properties — Tag Generation

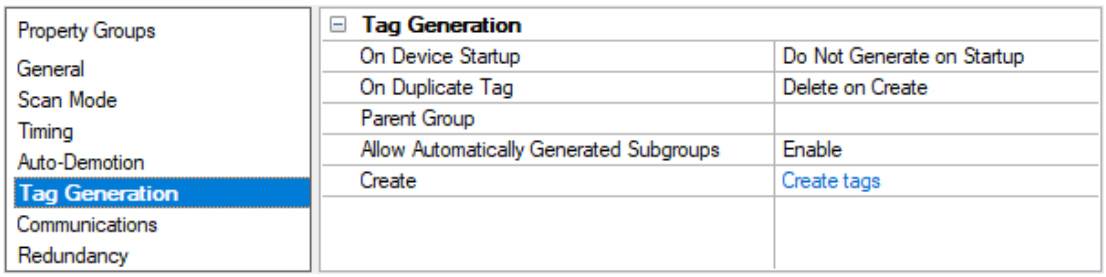
The automatic tag database generation features make setting up an application a plug-and-play operation. Select communications drivers can be configured to automatically build a list of tags that correspond to device-specific data. These automatically generated tags (which depend on the nature of the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

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

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

1. If a data acquisition system supports its own local tag database, the communications driver uses the tag names found in the device to build the server's tags.
2. If an Ethernet I/O system supports detection of its own available I/O module types, the communications driver automatically generates tags in the server that are based on the types of I/O modules plugged into the Ethernet I/O rack.

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




**On Property Change**: If the device supports automatic tag generation when certain properties change, the **On Property Change** option is shown. It is set to **Yes** by default, but it can be set to **No** to control over when tag generation is performed. In this case, the **Create tags** action must be manually invoked to perform tag generation. To invoke via the Configuration API service, access /config/v1/project/channels/{name}/devices/{name}/services/TagGeneration.

**On Device Startup**: Specify when OPC tags are automatically generated. Descriptions of the options are as follows:

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

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

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

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

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

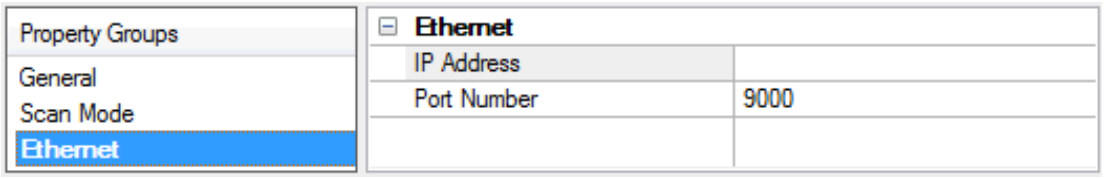
**Note:** Removing OPC tags affects tags that have been automatically generated by the communications driver as well as any tags that have been added using names that match generated tags. Users should avoid adding tags to the server using names that may match tags that are automatically generated by the driver.

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

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

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

## Device Properties — Ethernet




**IP Address**: Specify the IP address of the device to poll.
**Note**: TCP/IP must be properly installed to use this driver with Ethernet devices. *For more information on setting up TCP/IP, refer to Windows documentation.*

**Port Number**: Specify the Ethernet port to be used when connecting to a remote terminal server. The default is 9000.

## Automatic Tag Database Generation

The Dataforth isoLynx Driver supports an automatic tag database generation feature that enables drivers to automatically create tags to access data. It queries the device for its configuration and then uses that information to build a tag database.

### OPC Server Configuration

Automatic tag database generation can be customized to fit the application's needs. The primary control options can be set during the Database Creation step of the Device Wizard or later by clicking **Device Properties** | **Database Creation**. For more information on these properties, refer to the server help documentation.

### Operation

Depending on the configuration, tag generation may start automatically when the server project starts or be initiated manually at some other time. The server's Event Log will show when the tag generation process started, any errors that occurred while the device's configuration was queried and when the process completed.

### Group and Tag Naming Conventions

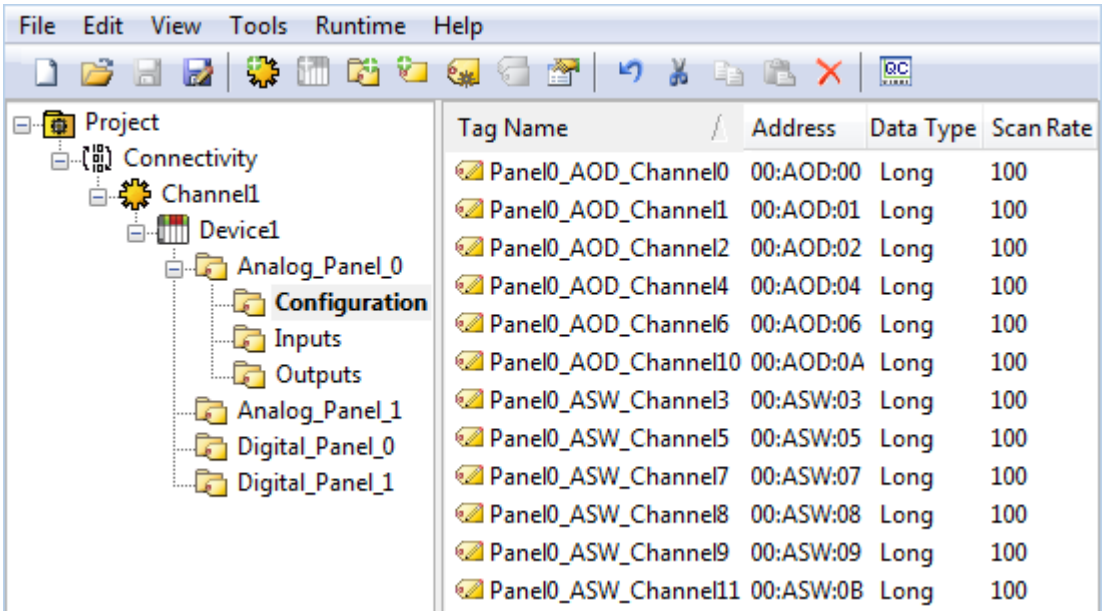
A group is created for each analog and digital panel in the system. Three subgroups are created under each panel's group for **Configuration**, **Inputs** and **Outputs** tags.

- The Configuration group contains tags that will be used for device configuration. Tag types include AOD, ASW, and DOD.
- The Inputs group contains tags for reading inputs. Tag types include AIA, AIC, and DI.
- The Outputs group contains tags for reading and writing to outputs. Tag types include AO and DO.

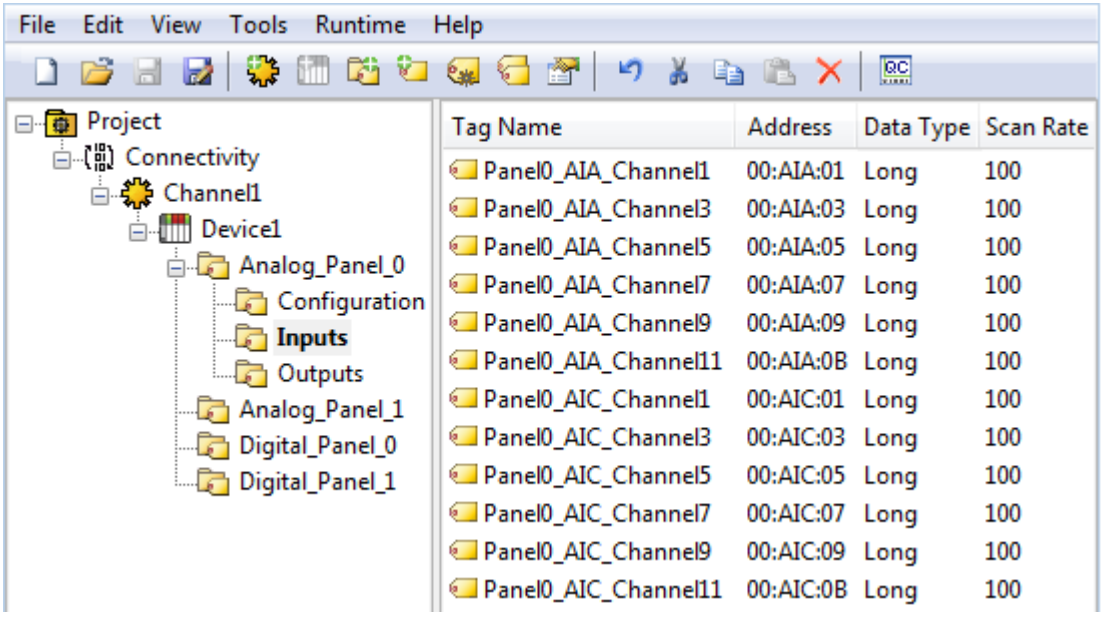
Each tag name includes the panel, tag type and channel number. The following images illustrate a system with two analog panels and two digital panels.

Digital panels names are numbered starting at 0 even though they are physically addressed at 8 and up.

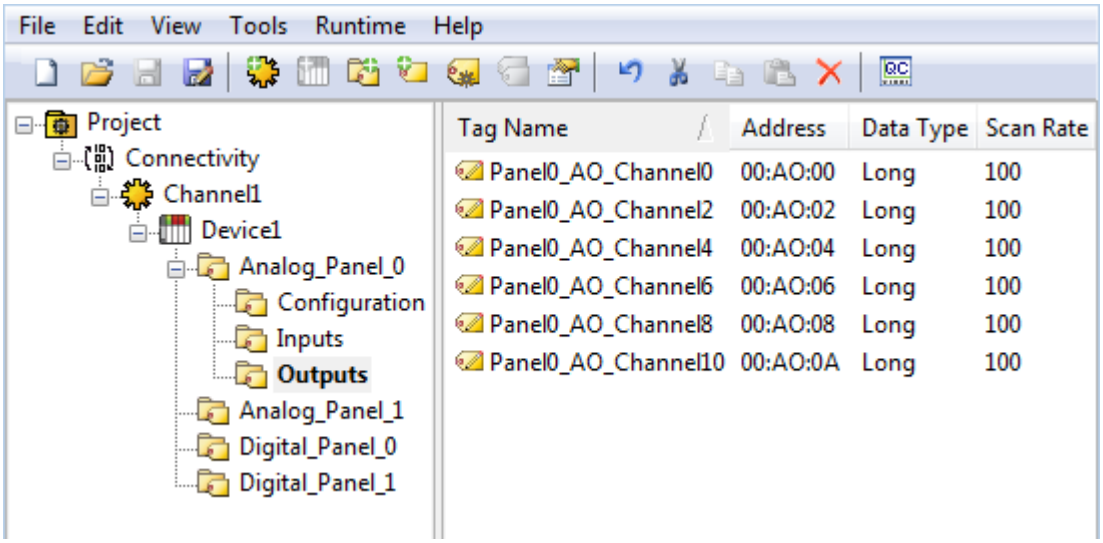
**Configuration**




**Inputs**




**Outputs**




## Data Types Description

| Data Type | Description |
|---|---|
| Boolean | Library returns a signed 32 bit value which is either a 0 or 1. |
| Long* | Library returns a signed 32 bit value. |

*Even though the data type is Long, all analog values are limited to a range of +32,767 to -32,768.

## Address Descriptions

The following table lists the address syntax for all of the supported addresses.

| Address | Description | Syntax | Data Type | Access |
|---|---|---|---|---|
| AIC | Analog Input Current | **PP:AIC:CC**<br>PP (panel) range: 00-03<br>CC (channel) range: 00-0F | Long* | Read Only |
| AIA | Analog Input Average | **PP:AIA:CC**<br>PP (panel) range: 00-03<br>CC (channel) range: 00-0F | Long* | Read Only |
| DI | Digital Input | **PP:DI:CC**<br>PP (panel) range: 08-0F<br>CC (channel) range: 00-0F | Boolean | Read Only |
| AO | Analog Output | **PP:AO:CC**<br>PP (panel) range: 00-03<br>CC (channel) range: 00-0F | Long* | Read/Write |
| ASW** | Analog Sample Weight | **PP:ASW:CC**<br>PP (panel) range: 00-03<br>CC (channel) range: 00-0F | Long* | Read/Write |
| DO | Digital Output | **PP:DO:CC**<br>PP (panel) range: 08-0F<br>CC (channel) range: 00-0F | Boolean | Read/Write |

| Address | Description | Syntax | Data Type | Access |
|---|---|---|---|---|
| AOD | Analog Output Default | **PP:AOD:CC**<br>PP (panel) range: 00-03<br>CC (channel) range: 00-0F | Long* | Read/Write |
| DOD | Digital Output Default | **PP:DOD:CC**<br>PP (panel) range: 08-0F<br>CC (channel) range: 00-0F | Boolean | Read/Write |

*Even though the data type is Long, all analog values are limited to range of +32,767 to -32,768.

**Tag type ASW (Analog Sample Weight) is a special case in that the value is limited to powers of 2 up to a maximum value of 16384. If an attempt is made to write a value that is not a power of 2, then the next power of 2 will be derived from that value and then written into the device. For example, if a value of 5 is written to an ASW tag, then 5 will be rounded up to 8 and then written to the device. Therefore, when reading back the value written, it will be 8 instead of 5. For more information, refer to section 3.3.4.9 "IOATTR_ISOLYNX_ AIOPTION_STRUCT" in the software user's manual.

**Notes:**

- Both panel and channel numbers are hexadecimal.
- Analog panel 0 ONLY has a channel range of 0 to B.
- The first digital panel's number would be 8 even though the physical address of the board would be likely set to 0. For more information, refer to section 6.0 "isoLynx Digital I/O Backpanel Description" in the hardware user's manual.

## Error Descriptions

The following error/warning messages may be generated. Click on the link for a description of the message.

### Address Validation

**Missing address**
**Device address '<address>' contains a syntax error**
**Address '<address>' is out of range for the specified device or register**
**Data Type '<type>' is not valid for device address '<address>'**
**Device address '<address>' is read only**

### Device Status Messages

**Device '<device name>' is not responding**
**Unable to write to '<address>' on device '<device name>'**

### Driver Error Messages

**Device '<device name>' responded with error '<Error Code>' (tag '<address>')**
**Device '<device name>' responded with error '<Error Code>' (tag '<address>') during connect**

### Automatic Tag Database Generation Messages

**Unable to Connect to Device or IO Inquire error**

**See Also**: Dataforth Data Acquisition Library Error Code Descriptions

## Dataforth Data Acquisition Library Error Code Descriptions

The following table lists the Error Codes returned by the Dataforth Data Acquisition Library.

| Error Code | Description |
|---|---|
| 0 | Success. |
| 1000 | Client already initialized. |
| 1001 | Failed to initialize client. |
| 1002 | Client not initialized. |
| 1003 | Logging not enabled. |
| 1004 | Failed to find error text. |
| 1005 | Invalid library link type. |
| 1006 | Invalid library callback type. |
| 1007 | Log file full. |
| 1008 | Memory pointer null. |
| 1009 | Failed to allocate memory. |
| 1010 | Failed to create thread. |
| 1011 | Invalid checksum type. |
| 2000 | Invalid communications processor type. |
| 2001 | Communications processor library does not exist. |
| 2002 | Failed to open communications processor device. |
| 2003 | Invalid communications processor device handle. |
| 2004 | Communications processor virtual function does not exist. |
| 2005 | Communications processor API function does not exist. |
| 2006 | Communications timeout. |
| 2007 | Communications cancelled. |
| 2008 | Invalid communications processor configuration type. |
| 2009 | Communications receive buffer pointer null. |
| 2010 | Communications send buffer pointer null. |
| 2011 | Communications send buffer empty. |
| 2012 | Non-blocking communications operation in progress. |
| 2012 | Communications port not initialized. |
| 2030 | Communications send operation with echo failed. |
| -2000 | Non-blocking communications operation pending. |
| -2020 | COM receive buffer overflow. |
| 3000 | Invalid I/O processor type. |
| 3001 | I/O processor library does not exist. |
| 3002 | Failed to open I/O processor device. |

| Error Code | Description |
|---|---|
| 3003 | Invalid I/O processor device handle. |
| 3004 | I/O processor virtual function does not exist. |
| 3005 | I/O processor API function does not exist. |
| 3006 | I/O timeout. |
| 3007 | I/O cancelled. |
| 3008 | Failed to add I/O device to list. |
| 3009 | Invalid I/O channel type. |
| 3010 | Invalid I/O channel list count. |
| 3011 | Invalid I/O channel panel. |
| 3012 | Invalid I/O channel number. |
| 3013 | I/O channel duplicate. |
| 3014 | I/O channel not configured. |
| 3015 | Invalid I/O channel group. |
| 3016 | Invalid I/O channel order. |
| 3017 | Failed to add I/O channel to list. |
| 3018 | Failed to parse I/O channel configuration. |
| 3019 | Invalid I/O channel attribute type. |
| 3020 | Invalid I/O channel attribute list count. |
| 3021 | Invalid I/O channel Read/Write control type. |
| 3022 | I/O processor function not implemented. |
| 3023 | Invalid I/O channel Read/Write samples. |
| 3024 | Invalid I/O channel Read/Write channel list count. |
| 3025 | Non-blocking I/O operation in progress. |
| 3026 | Invalid I/O processor command state. |
| 3027 | Invalid I/O response. |
| 3028 | Invalid I/O response length. |
| 3029 | Invalid I/O response checksum or CRC. |
| 3030 | I/O command not acknowledged. |
| -3000 | Non-blocking I/O operation pending. |
| 3100 | Invalid analog I/O range. |
| 5000 | Invalid date type. |
| 5001 | Invalid date string. |
| 6000 | String not found in file. |
| 6001 | Value not found in file. |
| 6002 | Error writing string to file. |
| 6003 | Error writing value to file. |
| -6000 | Maximum number of files exceeded. |
| 11000 | Failed to initialize serial COM port. |
| 11001 | Failed to open serial COM port. |

| Error Code | Description |
|---|---|
| 11002 | Invalid serial COM port. |
| 11003 | Serial COM port not present. |
| 11004 | Serial COM port already in use. |
| 11005 | Invalid serial COM IRQ. |
| 11006 | Invalid serial COM flow control. |
| 11007 | Invalid serial COM parity. |
| 12000 | Failed to initialize socket COM port. |
| 12001 | Failed to open socket COM port. |
| 12002 | Invalid socket COM port. |
| 12003 | Failed to connect to socket COM server. |
| 21000 | Invalid isoLynx script. |
| 21001 | Invalid isoLynx address. |
| 21002 | Invalid isoLynx panel. |
| 21020 | Invalid isoLynx I/F type. |
| 21021 | Invalid isoLynx I/F options. |
| 21022 | Invalid isoLynx I/F baudrate. |
| 21030 | Invalid isoLynx analog input range. |
| 21031 | Invalid isoLynx analog input average weight. |
| 21040 | Invalid isoLynx analog output range. |
| 21041 | Invalid isoLynx analog output initial data. |
| 21060 | Invalid isoLynx digital output initial data. |
| 21100 | isoLynx I/O command not acknowledged. |
| 21101 | isoLynx I/O command not acknowledged-Error Code 01. |
| 21102 | isoLynx I/O command not acknowledged-Error Code 02. |
| 21103 | isoLynx I/O command not acknowledged-Error Code 03. |
| 21104 | isoLynx I/O command not acknowledged-Error Code 04. |
| 21105 | isoLynx I/O command not acknowledged-Error Code 05. |
| 21106 | isoLynx I/O command not acknowledged-Error Code 06. |
| 21107 | isoLynx I/O command not acknowledged-Error Code 07. |
| 21108 | isoLynx I/O command not acknowledged-Error Code 08. |
| 21109 | isoLynx I/O command not acknowledged-Error Code 09. |
| 21116 | isoLynx I/O command not acknowledged-Error Code 10. |
| 21117 | isoLynx I/O command not acknowledged-Error Code 11. |
| 21118 | isoLynx I/O command not acknowledged-Error Code 12. |
| 21119 | isoLynx I/O command not acknowledged-Error Code 13. |
| 21120 | isoLynx I/O command not acknowledged-Error Code 14. |
| 21121 | isoLynx I/O command not acknowledged-Error Code 15. |
| 21122 | isoLynx I/O command not acknowledged-Error Code 16. |
| 21123 | isoLynx I/O command not acknowledged-Error Code 17. |
| 21200 | Invalid isoLynx IP address. |

| Error Code | Description |
|---|---|
| 21201 | Invalid isoLynx subnet mask. |
| 21202 | Invalid isoLynx gateway. |
| 21203 | Invalid isoLynx DNS server. |
| 21204 | Invalid keep alive timeout value. |
| -21000 | isoLynx I/O configuration does not match script. |
| -21001 | isoLynx I/F configuration does not match script. |
| -21030 | isoLynx analog input options do not match script. |
| -21040 | isoLynx analog output options do not match script. |
| -21050 | isoLynx digital input options do not match script. |
| -21060 | isoLynx digital output options do not match script. |
| -21200 | isoLynx network options do not match script. |

## Missing address

### Error Type:
Warning

### Possible Cause:
A tag address that has been specified dynamically has no length.

### Solution:
Re-enter the address in the client application.

## Device address '<address>' contains a syntax error

### Error Type:
Warning

### Possible Cause:
A tag address that has been specified dynamically contains one or more invalid characters.

### Solution:
Re-enter the address in the client application.

## Address <address>' is out of range for the specified device or register

### Error Type:
Warning

### Possible Cause:
A tag address that has been specified dynamically references a location that is beyond the range of supported locations for the device.

### Solution:

Verify that the address is correct; if it is not, re-enter it in the client application.

## Data Type '<type>' is not valid for device address '<address>'

### Error Type:
Warning

### Possible Cause:
A tag address that has been specified dynamically has been assigned an invalid data type.

### Solution:
Modify the requested data type in the client application.

## Device address '<address>' is Read Only

### Error Type:
Warning

### Possible Cause:
A tag address that has been specified dynamically has a requested access mode that is not compatible with what the device supports for that address.

### Solution:
Change the access mode in the client application.

## Device '<Device name>' is not responding

### Error Type:
Serious

### Possible Cause:
1. The connection between the device and the Host PC is broken.
2. The IP address or ID assigned to the device is incorrect.
3. The response from the device took longer to receive than the amount of time specified in the "Request Timeout" device property.
4. The interface type, com port or baud rate on device are configured incorrectly.

### Solution:
1. Verify the cabling between the PC and the device.
2. Verify the IP address or ID given to the named device matches that of the actual device.
3. Increase the Request Timeout property so that the entire response can be handled.

4. Verify that the Communications Channel Properties match those used when the device was configured with Dataforth configuration utility. For more information, refer to the section on "Communication Interface Reset Jumper" in the hardware user's manual.
5. Cycle power to device.

## Unable to write to '<address>' on device '<device name>'

### Error Type:
Serious

### Possible Cause:

1. The connection between the device and the Host PC is broken.
2. The IP address or ID assigned to the device is incorrect.
3. The interface type, com port or baud rate on device are configured incorrectly.

### Solution:

1. Verify the cabling between the PC and the device.
2. Verify the IP address or ID given to the named device matches that of the actual device.
3. Verify that the Communications Channel Properties match those used when the device was configured with Dataforth configuration utility. For more information, refer to the section on "Communication Interface Reset Jumper" in the hardware user's manual.

## Device '<device name>' responded with error 'Error Code' (Tag 'address')

### Error Type:
Serious

### Possible Cause:

1. The connection between the device and the Host PC is broken.
2. The IP address or ID assigned to the device is incorrect.
3. The interface type, com port or baud rate on device are configured incorrectly.

### Solution:

1. Verify the cabling between the PC and the device.
2. Verify the IP address or ID given to the named device matches that of the actual device.
3. Verify that the Communications Channel Properties match those used when the device was configured with Dataforth configuration utility. For more information, refer to the section on "Communication Interface Reset Jumper" in the hardware user's manual.

### Note:
The error code detailed in the message was returned by the Dataforth Data Acquisition Library . For more information, refer to **Dataforth Data Acquisition Library Error Code Descriptions**.

## Device '<device name>' responded with error 'Error Code' (Tag 'address') during Connect

### Error Type:
Serious

### Possible Cause:

1. The connection between the device and the Host PC is broken.
2. The IP address or ID assigned to the device is incorrect.
3. The interface type, com port or baud rate on device are configured incorrectly.

### Solution:

1. Verify the cabling between the PC and the device.
2. Verify the IP address or ID given to the named device matches that of the actual device.
3. Verify that the Communications Channel Properties match those used when the device was configured with Dataforth configuration utility. For more information, refer to the section on "Communication Interface Reset Jumper" in the hardware user's manual.

### Note:
The error code detailed in the message was returned by the Dataforth Data Acquisition Library while trying to connect to the device. For more information, refer to **Dataforth Data Acquisition Library Error Code Descriptions**.

## Unable to Connect to Device or IO Inquire error during tag Database Creation

### Error Type:
Serious

### Possible Cause:

1. The connection between the device and the Host PC is broken.
2. The IP address or ID assigned to the device is incorrect.
3. The interface type, com port or baud rate on device are configured incorrectly.
4. The I/O configuration set with Dataforth configuration utility is incorrect.

### Solution:

1. Verify the cabling between the PC and the device.
2. Verify the IP address or ID given to the named device matches that of the actual device.
3. Verify that the Communications Channel Properties match those used when the device was configured with Dataforth configuration utility. For more information, refer to the section on "Communication Interface Reset Jumper" in the hardware user's manual.
4. Run the Dataforth configuration utility and verify that the I/O configuration matches the physical hardware.

# Index

## A



## C



## D

## E



## G



## I



## M



## N

## O



## P



## R



## S



## T



## U

## W