Beckhoff TwinCAT Driver

Table of Contents

Beckhoff TwinCAT Driver

Help version 1.070

CONTENTS

Overview

The Beckhoff TwinCAT Driver is specifically designed to communicate with a Beckhoff TwinCAT software system. The Beckhoff TwinCAT software system turns any compatible PC into a real-time controller with a multi-PLC system, NC axis control, programming environment, and operating station. TwinCAT replaces conventional PLC and NC/CNC controllers (as well as operating devices) with the following:

- open, compatible PC hardware.
- embedded IEC 61131-3 software PLC, software NC and software CNC in Windows NT/XP/Vista, NT/ XP Embedded, and CE.
- programming and Runtime systems together on one PC or separated.
- connection to all common field buses.
- support of PC interfaces.
- data communication with user interfaces and other programs by means of open standards (such as OPC, OCX, DLL, and so forth).

World-wide Connection Through Message Routing

TwinCAT PLC programs can run on PCs or on Beckhoff Bus Terminal Controllers. A message router manages and distributes all the messages, both in the system and via TCP/IP connections. PC systems can be connected with each other via TCP/IP; Bus Terminal Controllers are integrated via serial interfaces and field buses.

TwinCAT ADS (Automation Device Specification)

The data link to TwinCAT servers occurs via the message router system, which enables Windows programs to both work with the local server and to exchange data with registered TwinCAT servers worldwide. The message router allows data exchange to remote servers on other PCs or field equipment.

Note: This driver has external dependencies. The Beckhoff TwinCAT ADS Communication Library is required on the same computer as the OPC server. To get the necessary files, download and install the free "Minimum Install" available from Beckhoff.

TwinCAT I/O – Universal I/O Interface For All Common Field Buses

Many PC fieldbus cards from various manufacturers are supported. It is possible to operate more than one fieldbus card per PC. Master and slave functionality is supported, depending on the selected fieldbus card. The fieldbus cards can be configured and diagnosed conveniently via the TwinCAT System Manager. TwinCAT I/O includes the TwinCAT real-time system for operating the fieldbuses and a DLL interface to application programs.

TwinCAT PLC

TwinCAT PLC was conceived as a pure software PLC. It allows up to four virtual PLC CPUs on one PC, with each running up to four separate user tasks. TwinCAT PLC includes both the programming environment and the Runtime system. Under the CE operating system and the embedded operating systems for the series BX and BC controllers, only TwinCAT Runtime is available. Program modifications are implemented via network-capable powerful communication with the Runtime system.

External Dependencies

This driver has external dependencies. The Beckhoff TwinCAT ADS Communication Library is required on the same computer as the server. To get the necessary files, download and install the free "Minimum Install" available from Beckhoff.

The TwinCAT Automation Device Specification (ADS) is a medium-independent protocol for transmissions within TwinCAT. As of this version, the files are available from the Beckhoff website under **Download | Software | TwinCAT 3 | TC1xxx | Runtime | TC1000 | TC3.1 ADS**.

Setup

Communication Protocol

ADS API

Supported Devices

Beckhoff TwinCAT PLC
BC9xxx Coupler Controller
BX9xxx Coupler Controller

Maximum Channels and Devices

The maximum number of channels supported by this driver is 100. The maximum number of devices per channel is 1024.

Device IDs

The Device ID is a specific ADS-AMS Net ID. Every PC on the network can be uniquely identified by a TCP/IP address. The ADS-AMS Net ID is an extension of the TCP/IP address and identifies a TwinCAT message router, such as "192.168.100.10.1.1". TwinCAT message routers exist on every TwinCAT PC and on every Beckhoff BCxxxx bus controller.

AMS Remote Connections Management

To communicate successfully with any remote TwinCAT runtime engine on a Beckhoff PLC (or an ADS library on any remote PC), the AMS Remote Connection Manager utility must be used to assign an AMS Net ID to the local station hosting the Beckhoff TwinCAT Driver. The AMS Net ID is also used to establish routes to remote AMS/ADS enabled devices. For more information, refer to **Setting Up an AMS Remote Connection**.

Important: On Windows Vista and above, User Account Control (UAC) must be turned off before a remote connection may be configured with the Beckhoff TwinCAT Remote Manager Utility. If the remote connection is created before UAC is disabled, the configuration settings will be created in an incorrect location (resulting in poor performance).

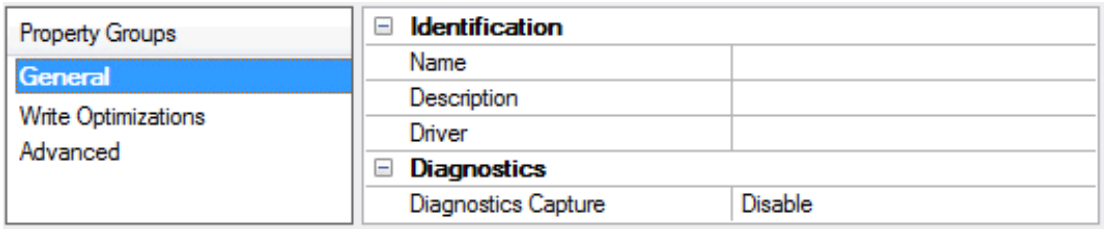
For more information about the AMS messaging protocol, consult Beckhoff's help documentation.

Tag Database Creation

The Automatic OPC Tag Database Generation features of this driver have been designed to make setting up the OPC application less time consuming. This driver can be configured to automatically build a list of server tags within the server that correspond to device specific data. The automatically generated OPC tags can then be browsed from the OPC client.

Channel Properties — General

This server supports the use of simultaneous multiple 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: 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: User-defined information about this channel.

Many of these properties, including Description, have an associated system tag.

Driver: Selected protocol / driver for this channel. This property specifies 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. With this in mind, changes to the properties should not be made once a large client application has been developed. Utilize the User Manager to prevent operators from changing properties and restrict access rights to server features.

Diagnostics

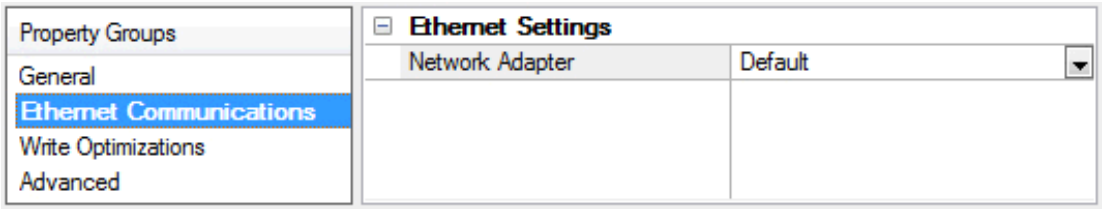
Diagnostics Capture: When enabled, this option makes the channel's diagnostic information available to OPC applications. Because the server's diagnostic features require a minimal amount of overhead processing, it is recommended that they be utilized when needed and disabled when not. The default is disabled.

Note: This property is not available if the driver does not support diagnostics.

For more information, refer to "Communication Diagnostics" and "Statistics Tags" in the server help.

Channel Properties — Ethernet Communications

Ethernet Communication can be used to communicate with devices.

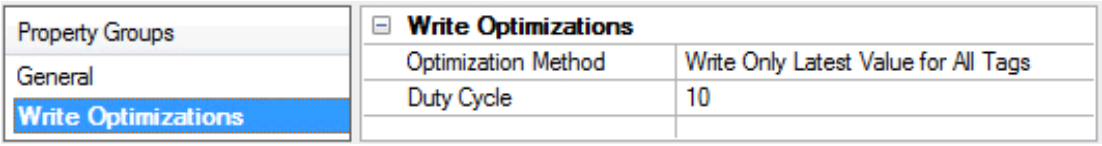


Ethernet Settings

Network Adapter: Specify the network adapter to bind. When left blank or Default is selected, the operating system selects the default adapter.

Channel Properties — Write Optimizations

As with any server, writing data to the device may be the application's most important aspect. The server intends to ensure that the data written from the client application gets to the device on time. Given this goal, the server provides optimization properties that can be used 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:

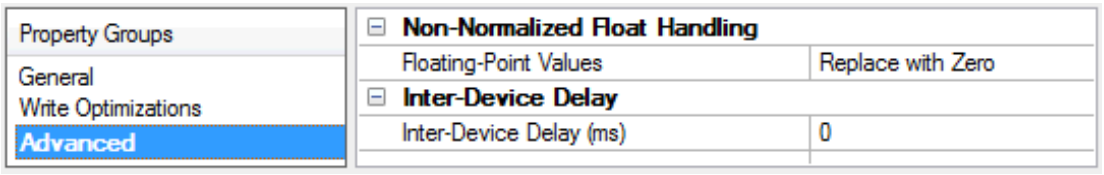
- **Write All Values for All Tags**: This option forces the server to attempt to write every value to the controller. In this mode, the server continues to gather write requests and add them to the server's internal write queue. The server processes the write queue and attempts to empty it by writing 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 not available 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.

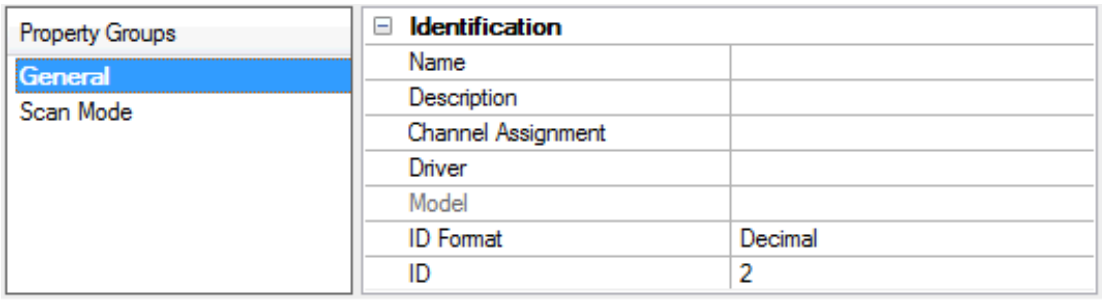
For more information on the floating point values, refer to "How To ... Work with Non-Normalized Floating Point Values" in the server help.

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: This property specifies 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: User-defined information about this device.
Many of these properties, including Description, have an associated system tag.

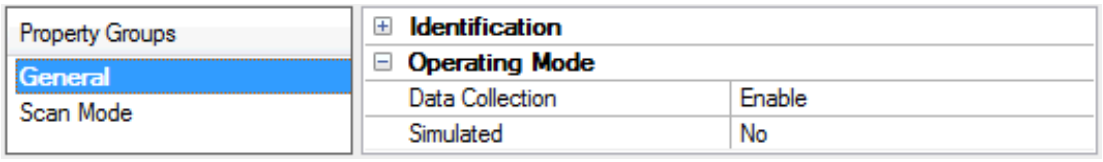
Channel Assignment: User-defined name of the channel to which this device currently belongs.

Driver: Selected protocol driver for this device.

Model: This property specifies the specific type of device that is associated with this ID. The contents of the drop-down menu depends 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 help documentation.

ID: This property specifies 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. *For more information, refer to the driver's help documentation.*

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: This option places the device into 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 item's memory map is based on the group Update Rate. The data is not saved if the server removes the item (such as when the server is reinitialized). The default is No.

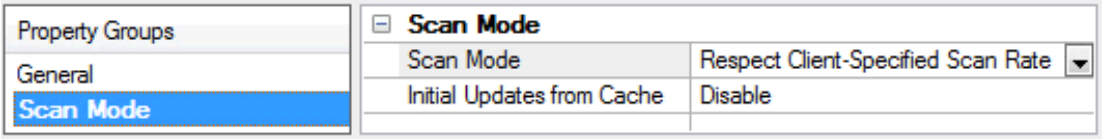
Notes:

1. This 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.
2. 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.

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

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: Specifies how tags in the device are scanned for updates sent to subscribing clients. Descriptions of the options are:

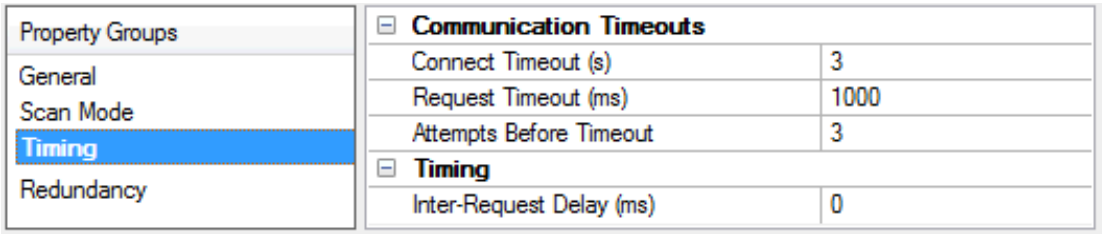
- **Respect Client-Specified Scan Rate**: This mode uses the scan rate requested by the client.
- **Request Data No Faster than Scan Rate**: This mode specifies the value set as the maximum scan rate. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
  Note: When the server 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 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: This property specifies an interval used by all drivers to determine how long the driver waits for a response from the target device to complete. The valid range is 50 to 9,999,999 milliseconds (167.6667 minutes). The default is usually 1000 milliseconds, but can vary depending on the driver. The default timeout for most serial drivers is based on a baud rate of 9600 baud or better. When using a driver at lower baud rates, increase the timeout to compensate for the increased time required to acquire data.

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

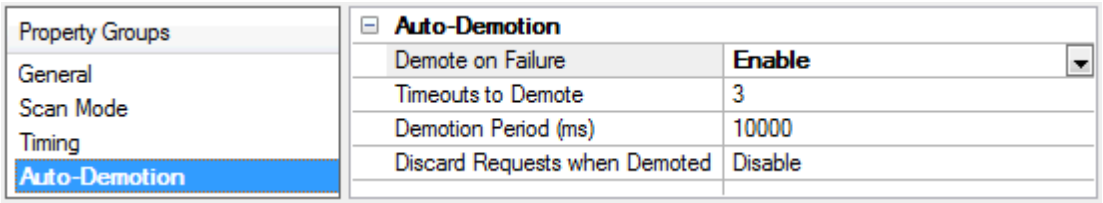
Timing

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

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

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

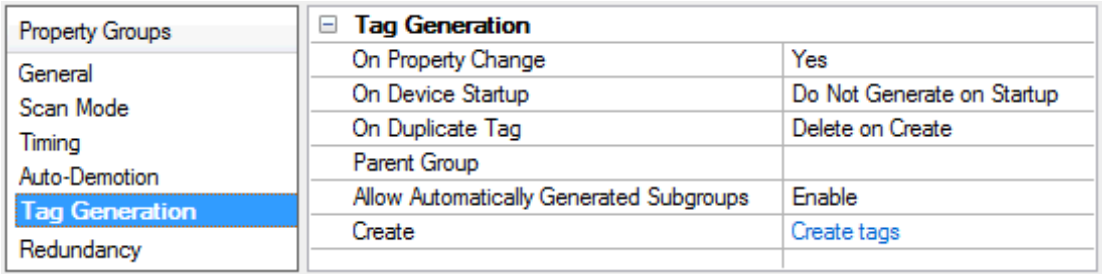
The automatic tag database generation features make setting up an application a plug-and-play operation. Select communications drivers can be configured to automatically build a list of tags that correspond to device-specific data. These automatically generated tags (which depend on the nature of the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

On Device Startup: This property specifies 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.

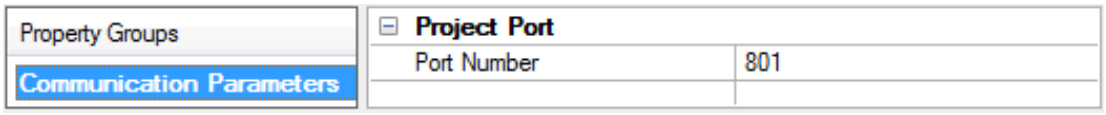
Note: Removing OPC tags affects tags that have been automatically generated by the communications driver as well as any tags that have been added using names that match generated tags. Users should avoid adding tags to the server using names that may match tags that are automatically generated by the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



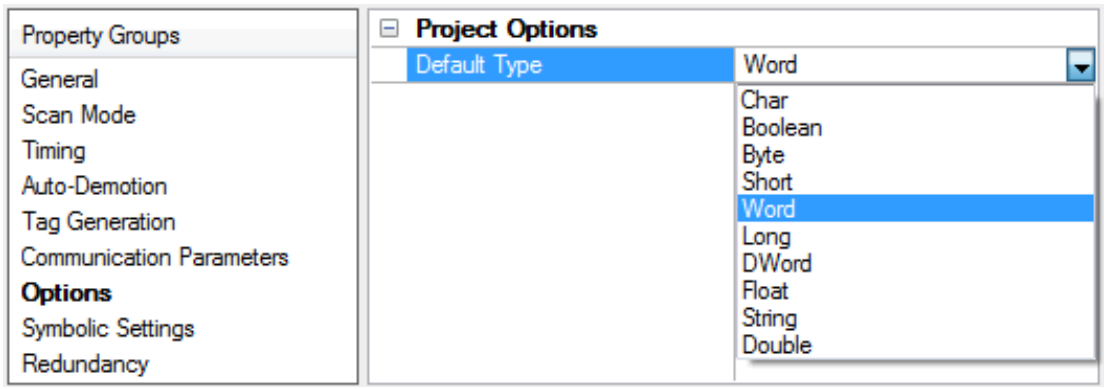
Project Port

Port Number
The ADS devices in a TwinCAT message router are uniquely identified by a number referred to as the ADS-PortNr. For more information, refer to table below.

System	Supported ADS-PortNr	Default Setting
TwinCAT 2 PLC Runtime Systems	801 811 821 831	801
TwinCAT 3 PLC Runtime Systems	851 852 853 854	851
BC9xxx/BX9xxx Series Controllers	800	800

Note: Users must allow the TCP port number 48898 (AMS port 801) on the firewall if planning to connect remotely to an ADS/AMS router. If not, the system will not work reliably and may stop.

Device Properties — Options

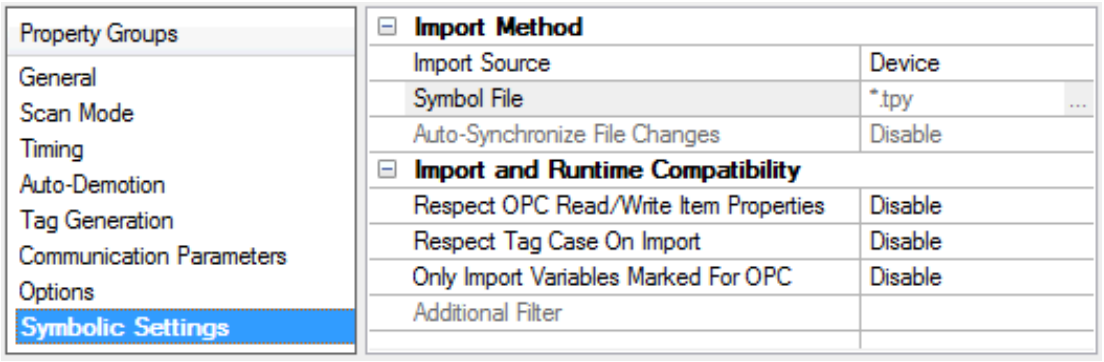


Project Options

Default Type: Specify the data type tol be assigned to a client/server tag when the default type is selected during tag addition, modification, and import. Client/server tags are assigned the default data type when any of the following conditions occur:

1. A Dynamic Tag is created in the client with Native as its assigned data type.
2. A Static Tag is created in the server with Default as its assigned data type.

Device Properties — Symbolic Settings



Import Method

The generated server tags are based on the tags defined in the Beckhoff TwinCAT device. There are two database import methods that can be used to create a tag database: Upload Symbols from Device and Upload Symbols from File.

- **Upload Symbols from Device:** This feature retrieves the tags directly from the controller over the same Ethernet connection used for data access.
  Note: At this time, tags and symbols can only be uploaded from a TwinCAT soft PLC. They cannot be uploaded from a hardware BC/BX controller.
- **Upload Symbols from File:** This feature requires that, when creating the tag database from an import file, the import file be a .tpy file generated from the Beckhoff TwinCAT PLC Control software.
- **Symbol File:** This property specifies the exact location of the .tpy file from which tags will be imported. BC9xxx and BX9xxx models must include this file for a Runtime database.
- **Auto-Synchronize File Changes:** This feature updates the Runtime database automatically when the file has been modified.
  This option is only supported by BC/BX model devices.
  Caution: Enabling Auto-Synchronize will reflect changes to the Runtime, regardless of whether or not the new configuration has been pushed down to the device. Users should ensure that the device is using the same .tpy file as loaded in the server project; otherwise, inaccurate data could be obtained.

Notes:

1. Tag database changes will not appear in the server until Automatic Tag Generation is performed.
2. The BC9xxx and BX9xxx models only support database creation from a .tpy file.

Import and Runtime Compatibility

Additional OPC item information is located in the comment field after the PLC variable declaration in TwinCAT PLC Control. This optional functionality is used to minimize the OPC namespace of the server project. It also assigns the appropriate read/write item properties and OPC client access.

Respect OPC Read/Write Item Properties
When enabled, variables that include the string "OPC_PROP[0005]: <r/w access number>" (defined within the variable comment field in the PLC project) will be assigned the appropriate Read/Write access. During import and Runtime, any tag(s) corresponding to a variable with the R/W access specified will carry the same Read/Write access rights. Whitespace is ignored.

- **Syntax:** *(*~(OPC_PROP[0005] : <Item Access Rights> : <remark>)*)*

  <Item Access Rights> 1 = Read Only, 2 = Write Only, 3 = Read/Write.
  <remark> Optional and ignored.

- **Read Only Example:** dwTemperature:DWORD; (*~(OPC_PROP[0005]:1:Read Only)*)

Respect Tag Case on Import
When disabled (default), imported tag and address strings are forced into upper case. When enabled, upper and lower case text is preserved when imported.

Note: Respect Tag Case on Import behavior:

- When disabled (default) and importing from a .tpy file, all text (names and addresses) are forced into upper case.
- When enabled and importing from a .tpy file, mixed-case text is preserved.
- When importing directly from a TwinCAT-2 PLC, all text is upper case, regardless of this setting.
- When enabled and importing directly from a TwinCAT-3 PLC, mixed-case text is preserved.
- When disabled (default) and importing directly from a TwinCAT-3 PLC, names contain mixed case, but addresses are in upper case.

Only Import Variables Marked for OPC
When enabled, only tags with the keyword "OPC:1" (defined within the variable comment field in the PLC project) will be imported and visible during Runtime.

- **Syntax:** *(*~(OPC : 1 : <remark>)*)*

  <remark> Optional and ignored.

- **Example:** bMemFlag AT%MX0.0:BOOL; (*~(OPC:1:Visible to OPC Clients)*)

Only Import Variables Marked for OPC + Filter
When Only Import Variables Marked for OPC is enabled with a Filter string, only tags in which the comment includes the string "OPC:1" and "OPC_Filter: <filter>" will be imported into the project and visible during Runtime. Filter strings are arbitrary, user-defined strings used for another layer of filtering. If none are specified, this field will be ignored. Otherwise, the variable will only be imported if the filter string found in the comment section matches the filter set in Device Properties. Whitespace is treated literally.

- **Syntax:** *(*~(OPC : 1 : <remark>)(OPC_Filter : <filter string> : <remark>)*)*

  <remark> Optional and ignored.

- **Example:** bMemFlag AT%MX0.0:BOOL; (*~(OPC:1:Visible to OPC Clients)(OPC_Filter:Memory Access)*)

Combination Usage

Each of the comments described above can be combined for greater control over the filtering and assignment of OPC Read/Write item properties.

Only Import Variables Marked for OPC + Respect OPC Read/Write Item Properties
Variables will only be imported and visible during Runtime if "OPC:1" is defined in the comment field. Upon being imported, corresponding tags will be assigned the appropriate Read/Write access rights.

- **Syntax:***(*~(OPC : 1 : <remark>)(OPC_PROP[0005] : <Item Access Rights> : <remark>)*)*
- **Example:** dwTemperature:DWORD; (*~(OPC:1:Visible to OPC Clients)(OPC_PROP[0005]:1:Read Only)*)

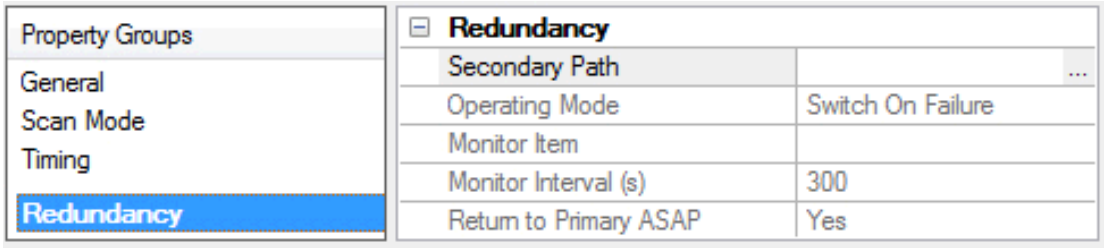
Only Import Variables Marked for OPC + Filter + Respect OPC Item Properties
A variable with this comment will only be imported and visible during Runtime if "OPC:1" is defined in the comment field and if the filter string found in the comment section matches the filter set in Device Properties. Corresponding tags will be assigned Read/Write access rights once they are imported.

- **Syntax:** *(*~(OPC : 1 : <remark>)(OPC_Filter : <filter string> : <remark>) (OPC_PROP[0005] : <Item Access Rights> : <remark>)*)*
- **Example:** dwTemperature:DWORD; (*~(OPC:1:Visible to OPC Clients) (OPC_Filter:Memory Access) (OPC_PROP[0005]:1:Read Only)*)

OPC Item Description + Only Import Variables Marked for OPC
Any characters placed before the tilde in the comment field are treated as a description and will be displayed in the server as an Item Description.

- **Syntax:***(*<OPC Item Description>~(OPC : 1 : <remark>)*)*
- **Example:** (*This is my OPC Item Description~(OPC :1:Visible to OPC Clients)*)

Device Properties — Redundancy



Redundancy is available with the Media-Level Redundancy Plug-In.
Consult the website, a sales representative, or the user manual for more information.

Automatic Tag Database Generation

Controller-to-Server Name Conversions

Leading underscores
Leading underscores (_) in tag/program names will be replaced with U_. This is required since the server does not accept tag/group names beginning with an underscore.

Preparing for Automatic Tag Database Generation

Upload Symbols from Device

It is recommended that all communications to the Beckhoff TwinCAT of interest are halted during the database creation process.

In the server

1. To start, open the **Device Properties** for the device in which tags will be generated.
2. Select the **Database Settings** tab, and then select **Create tag database from device**.

3. Next, select the desired **Beckhoff Compatibility** options.
4. Then, select the **Options** tab and make the desired changes.
5. Select the **Database Creation** tab and utilize as instructed in **Database Creation Settings**.

Note: At this time, tags and symbols can only be uploaded from a TwinCAT soft PLC. They cannot be uploaded from a hardware BC/BX controller.

Upload Symbols from File

The driver uses a file generated from Beckhoff TwinCAT PLC called an .tpy import/export file to generate the tag database.

Note: All tags, including global and program, are imported and expanded according to their data type.

In Beckhoff TwinCAT PLC

The .tpy file is automatically created the first time that a build is done on a clean project.

1. Open the **TwinCAT PLC Control** program.
2. If using a TwinCAT 2 PLC, click **Project** | **Clean All**. If using a TwinCAT 3 PLC, click **Build** | **Clean Solution**.
3. Then, if using a TwinCAT 2 PLC, click **Project** | **Build**. If using a TwinCAT 3 PLC, click **Build** | **Build Solution**.

Notes:

1. The program does not generate the .tpy file again until the project is cleaned; however, the .tpy always regenerates when the project is rebuilt.
2. The .tpy file is located in the same directory as the PLC project.

In the server

1. To start, open the **Device Properties** for the device in which tags will be generated.
2. Select the **Database Settings** tab, and then select **Create tag database from import file**.
3. Enter or browse for the location of the .tpy file that was previously created.
4. Next, select the desired **Beckhoff Compatibility** options.
5. Then, select the **Options** tab and make the desired changes.
6. Select the **Database Creation** tab and utilize as instructed in **Database Creation Settings**.

Note: Online tag generation for PLC projects that contain Step-Transitions produce Boolean tags for each step, describing the state as active or inactive. Offline generation does not produce Step-Transition tags.

Performance Optimization

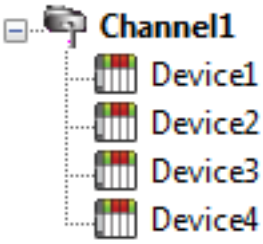
Optimizing Communications

With any programmable controller there are unique ways for optimizing system throughput, and the Beckhoff TwinCAT Driver is no different from the rest. The Beckhoff TwinCAT Driver is designed to optimize reads and writes. For tags of all data types, requests are grouped into a single transaction. This provides

drastic improvement in performance over single tag transaction. The only limitation is on the number of tags that can fit in a single transaction.

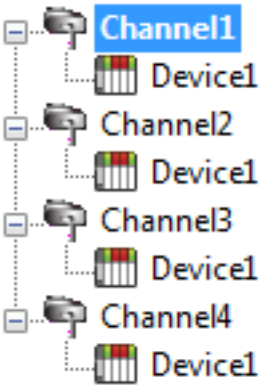
Optimizing the Application

While the driver is fast, there are a couple of guidelines that can be used to control and optimize the application and gain maximum performance. The server refers to communications protocols like Beckhoff TwinCAT as a channel. Each channel defined in the application represents a separate path of execution in the server. Once a channel has been defined, a series of devices may then be defined under that channel. Each of these devices represents a single Beckhoff TwinCAT controller from which data will be collected. While this approach to defining the application will provide a high level of performance, it won't take full advantage of the driver or the network. An example of how the application may appear when configured using a single channel is shown below.



In this example, each device appears under a single channel called Beckhoff_ TwinCAT. In this configuration, the driver must move from one device to the next as quickly as possible to gather information at an effective rate. As more devices are added or more information is requested from a single device, however, the overall update rate begins to suffer.

If the Beckhoff TwinCAT Driver could only define one single channel, then the example shown above would be the only option available; however, the driver can define up to 100 channels. Using multiple channels distributes the data collection workload by simultaneously issuing multiple requests to the network. An example of how the same application may appear when configured using multiple channels to improve performance is shown below.



Each device has now been defined under its own channel. In this new configuration, a single path of execution is dedicated to the task of gathering data from each device. If the application has 100 or fewer devices, it can be optimized exactly how it is shown here.

The performance will improve even if the application has more than 100 devices. While 100 or fewer devices may be ideal, the application will still benefit from additional channels. Although by spreading the device load across all channels will cause the server to move from device to device again, it can now do so with far fewer devices to process on a single channel.

Data Type Descriptions

Data Type	Description
Boolean	Single bit
Byte	Unsigned 8-bit value
Char	Signed 8-bit value
Word	Unsigned 16-bit value
Short	Signed 16-bit value

Data Type	Description
DWord	Unsigned 32-bit value
Long	Signed 32-bit value
Float	32-bit IEEE Floating point
Double	64-bit IEEE Floating point
String	Null terminated character array

*For a description of TwinCAT-platform specific data types, refer to **TwinCAT Data Types**.*

Unsupported Data Types

Unsupported data types include LBCD and BCD.

Address Descriptions

The following table summarizes the valid addressing formats in the server.

Address Format	Notation	Example	Notes
Standard	<tag name>	tag_1	Tag cannot be an array
Array Element	<array tag name> [dim1, dim2, dim3]	tag_1 [2, 58, 547] tag_1 [0,3]	Dimension Range=1 to 3 Element Range=0 to 65535
String	<tag name>/<string length>	tag_1/4	Length Range=1 to 65535 The number of characters to read/write equals the string length.

*For information on how elements are referenced for 1, 2 and 3 dimensional arrays, refer to **Ordering of Array Data**.*

See Also** **Addressing Atomic Data Types

Symbolic Tag Based Addressing

The Beckhoff TwinCAT Driver uses a tag or symbol-based addressing structure. These tags (which are commonly referred to as Native Tags) differ from conventional PLC data items in that the tag name itself is the address, not a physical or logical address.

Client/Server Tag Address Rules

Beckhoff TwinCAT variable names correspond to Client/Server Tag addresses. Beckhoff TwinCAT variable names (entered via Beckhoff TwinCAT PLC) follow the IEC 61131-3 identifier rules. Client/Server Tag addresses follow these same rules and are listed below:

- Must begin with an alphabetic (A-Z, a-z) character or an underscore (_).
- Can only contain alphanumeric characters and underscores.
- First 32 characters are significant.

- Cannot have consecutive underscores.
- Are not case sensitive.

Client/Server Tag Name Rules

The rules for tag name assignment in the server differs from address assignment because tag names cannot begin with an underscore.

Tag Scope

Global tags are Beckhoff TwinCAT variables that have global scope in the controller. Any program or task can access the global tags, which use the following notation:

. <tag name>

Program Tags

Program tags are identical to global tags, except that a program tag's scope is local to the program it is defined in. Program tags follow the same addressing rules and limitations as global tags. The only difference is that program tags are prefixed with the following notation:

<program name> . <tag name>

For example, Beckhoff TwinCAT variable "tag_1" in program "prog_1" would be addressed as "prog_1.tag_1" in a Client/Server Tag address.

Structure Tag Addressing

Beckhoff TwinCAT structure variables, global or program, contain one or more member variables. Member variables can be atomic or structured in nature.

Global tag example:	Program tag example:
.<structure name>.<atomic-type tag>	<program name>.<structure name>.<atomic-type tag>

This implies that a substructure would be addressed as:

Global Tag Example	Program tag example:
.<structure name>.<substructure name>.<atomic-type tag>	<program name>.<structure name>.<substructure name>.<atomic-type tag>

Arrays of structures would be addressed as:

Global Tag Example	Program tag example:
.<structure array name>[dim1, dim2, dim3].<atomic-type tag>	<program name>.<structure array name>[dim1, dim2, dim3].<atomic-type tag>

Again, this implies that an array of substructures would be addressed as:

Global Tag Example	Program tag example:
.<structure name>.<substructure array name>[dim1, dim2, dim3].<atomic-type	<program name>.<structure name>.<substructure array name>[dim1, dim2, dim3].<atomic-type tag>

tag>	

Note: These are a few of the many addressing possibilities involving structures. These are shown here only to provide an introduction to structure addressing. For more information, refer to the Beckhoff TwinCAT documentation.

Addressing Atomic Data Types

The table below shows suggested usages and addressing possibilities for a TwinCAT Data Type given the address formats available.

Atomic Data Type	Client/Server	Standard	Array Element	String
BOOL	Data Type Tag Example	Boolean BOOLTAG	Boolean BOOLARR[0]	
SINT/USINT	Data Type Tag Example	Byte, Char SINTTAG	Byte, Char SINTARR[0]	String SINTARR[0]/4 String length cannot be longer than the array length minus the element position.
INT/UINT	Data Type Tag Example	Word, Short INTTAG	Word, Short INTARR[0]	
DINT/UDINT	Data Type Tag Example	DWord, Long DINTTAG	DWord, Long DINTARR[0]	
REAL	Data Type Tag Example	Float REALTAG	Float REALARR[0]	
LREAL*	Data Type Tag Example	Double LREALTAG	Double LREALARR[0]	
TIME	Data Type Tag Example	DWord, Long TIMETAG	DWord, Long TIMEARR[0]	
STRING	Data Type Tag Example	String STRINGTAG/80	String STRINGARR[1]/80	

*This data type is not supported by the BC/BX Controller model.

Unsupported TwinCAT Data Types

Unsupported data types include pointers, enumerations, TOD, and DT.

See Also: *Address Descriptions*

Ordering of TwinCAT Array Data

1. Dimensional Arrays - array [dim1]
1 dimensional array data is passed to and from the controller in ascending order:

for (dim1=0; dim1<dim1_max; dim1++)

Example: 3 element array

array [0] array [1] array [2]

2. Dimensional Arrays - array [dim1, dim2]
2 dimensional array data is passed to and from the controller in ascending order:

for (dim1=0; dim1<dim1_max; dim1++)
for (dim2=0; dim2<dim2_max; dim2++)

Example: 3X3 element array
array [0, 0]
array [0, 1]
array [0, 2]
array [1, 0]
array [1, 1]
array [1, 2]
array [2, 0]
array [2, 1]
array [2, 2]

3. Dimensional Arrays - array [dim1, dim2, dim3]
3 dimensional array data is passed to and from the controller in ascending order:

for (dim1 = 0; dim1 < dim1_max; dim1++)
for (dim2 = 0; dim2 < dim2_max; dim2++)
for (dim3 = 0; dim3 < dim3_max; dim3++)

Example: 3X3x3 element array
array [0, 0, 0]
array [0, 0, 1]
array [0, 0, 2]
array [0, 1, 0]
array [0, 1, 1]
array [0, 1, 2]
array [0, 2, 0]
array [0, 2, 1]
array [0, 2, 2]
array [1, 0, 0]
array [1, 0, 1]
array [1, 0, 2]
array [1, 1, 0]
array [1, 1, 1]
array [1, 1, 2]
array [1, 2, 0]
array [1, 2, 1]
array [1, 2, 2]

array [2, 0, 0]
array [2, 0, 1]
array [2, 0, 2]
array [2, 1, 0]
array [2, 1, 1]
array [2, 1, 2]
array [2, 2, 0]
array [2, 2, 1]
array [2, 2, 2]

Error Descriptions

The following categories of messages may be generated. Click on a link for a list of messages.

Address Validation
Automatic Tag Database Generation Error Messages
Driver Error Messages
Device Status Messages
Read Error Messages
Write Error Messages

Error Codes

The following sections define error codes that may be encountered in the event log of the server. Refer to the Event Log section within the Server Options chapter of the server help file for detailed information on how the event logger works.

Hex	Dec	Description
0x000	0	No error.
0x001	1	Internal error.
0x002	2	No Runtime.
0x003	3	Allocation locked memory error.
0x004	4	Insert mailbox error.
0x005	5	Wrong receive HMSG.
0x006	6	Target port not found.
0x007	7	Target machine not found.
0x008	8	Unknown Command ID.
0x009	9	Bad Task ID.
0x00A	10	No IO.
0x00B	11	Unknown AMS command.
0x00C	12	Win 32 error.
0x00D	13	Port is not connected.
0x00E	14	Invalid AMS length.
0x00F	15	Invalid AMS Net ID.
0x010	16	Low installation level.
0x011	17	No debug available.

Hex	Dec	Description
0x012	18	Port is disabled.
0x013	19	Port is already connected.
0x014	20	AMS Sync Win32 error.
0x015	21	AMS Sync Timeout.
0x016	22	AMS Sync AMS error.
0x017	23	AMS Sync no index map.
0x018	24	Invalid AMS port.
0x019	25	No memory.
0x01A	26	TCP send error.
0x01B	27	Host is unreachable.
0x500	1280	ROUTERERR_NOLOCKEDMEMORY No locked memory can be allocated.
0x501	1281	ROUTERERR_RESIZEMEMORY The size of the router memory could not be changed.
0x502	1282	ROUTERERR_MAILBOXFULL The Mailbox of the AMS Port has reached the maximum number of possible entries. The current sent message was abolished.
0x503	1283	ROUTERERR_DEBUGBOXFULL The mailbox of the AMS debugger port has reached the maximum number of possible entries. The sent message will not be displayed in the debug monitor.
0x504	1284	ROUTERERR_UNKNOWNPORTTYPE
0x505	1285	ROUTERERR_NOTINITIALIZED Router is not yet initialized.
0x506	1286	ROUTERERR_PORTALREADYINUSE The desired port number is already assigned.
0x507	1287	ROUTERERR_NOTREGISTERED The AMS Port is not registered.
0x508	1288	ROUTERERR_NOMOREQUEUES The maximum number of AMS Ports has been reached.
0x509	1289	ROUTERERR_INVALIDPORT
0x50A	1290	ROUTERERR_NOTACTIVATED The TwinCAT Router is not yet active.
0x700	1792	Error class <device error>.
0x701	1793	Service is not supported by the server.
0x702	1794	Invalid index group.
0x703	1795	Invalid index offset.
0x704	1796	Reading/writing is not permitted.
0x705	1797	Parameter size is not correct.
0x706	1798	Invalid parameter value(s).
0x707	1799	Device is not in a ready state.
0x708	1800	Device is busy.

Hex	Dec	Description
0x709	1801	Invalid context (must be in Windows).
0x70A	1802	Out of memory.
0x70B	1803	Invalid parameter value(s).
0x70C	1804	Not found (files and etc.).
0x70D	1805	Syntax error in command or file.
0x70E	1806	Objects do not match.
0x70F	1807	Object already exists.
0x710	1808	Symbol not found.
0x711	1809	Symbol version is invalid.
0x712	1810	Server is in invalid state.
0x713	1811	AdsTransMode is not supported.
0x714	1812	Notification handle is invalid.
0x715	1813	Notification client not registered.
0x716	1814	No more notification handles.
0x717	1815	Size for watch too big.
0x718	1816	Device is not initialized.
0x719	1817	Device has a timeout.
0x71A	1818	Query interface failed.
0x71B	1819	Wrong interface required.
0x71C	1820	Class ID is invalid.
0x71D	1821	Object ID is invalid.
0x71E	1822	Request is pending.
0x71F	1823	Request is aborted.
0x720	1824	Signal warning.
0x721	1825	Invalid array index.
0x722	1826	The symbol is not active. Release the handle and try again.
0x723	1827	Access is denied.
0x740	1856	Error class <client error>.
0x741	1857	Invalid parameter at service.
0x742	1858	Polling list is empty.
0x743	1859	Var connection already in use.
0x744	1860	Invoke ID in use.
0x745	1861	Timeout elapsed.
0x746	1862	Error in win32 subsystem.
0x748	1864	Ads-port not opened.
0x750	1872	Internal error in ads sync.
0x751	1873	Hash table overflow.
0x752	1874	Key not found in hash.
0x753	1875	No more symbols in cache.

Hex	Dec	Description
0x754	1876	An invalid response was received.
0x755	1877	The sync port is locked.
0x274c	10060	A connection attempt failed because the connected party did not properly respond after a period of time, or the established connection failed because the connected host has failed to respond.
0x274d	10061	No connection could be made because the target machine actively refused it.
0x2751	10065	A socket operation was attempted to an unreachable host.

Address Validation

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

Address <address> is out of range for the specified device or register.
Array size is out of range for address <address>.
Data type <type> is not valid for device address <address>.
Device Address <address> contains a syntax error.

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 this device.

Solution:

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

Array size is out of range for address <address>.

Error Type:

Warning

Possible Cause:

A tag address that has been specified statically is requesting an array size that is too large.

Solution:

Re-enter the address in the client application to specify a smaller value for the array or a different starting point.

Data type <type> is not valid for device address <address>.

Error Type:

Warning

Possible Cause:

A tag address that has been specified statically has been assigned an invalid data type.

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

Device Address <address> contains a syntax error.

Error Type:
Warning

Possible Cause:
A tag address that has been specified statically contains one or more of the following errors:

1. The address doesn't conform to the tag address naming conventions.
2. The address is invalid according to the address format and underlying controller tag data type.
3. A program tag was specified incorrectly.
4. The address used an invalid format.

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

Automatic Tag Database Generation Error Messages

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

Unable to generate a tag database for device <device name>. Reason: Beckhoff TwinCAT ADS DLL necessary for import is not loaded.
Unable to generate a tag database for device <device name>. Reason: Device is not responding.
Unable to generate a tag database for device <device name>. Reason: Device returned error code <#>.
Unable to generate a tag database for device <device name>. Reason: Memory allocation error.

Unable to generate a tag database for device <device name>. Reason: Beckhoff TwinCAT ADS DLL necessary for import is not loaded.

Error Type:
Warning

Possible Cause:
The driver was unable to load the TcAdsDll.dll, which is necessary for communications with the Beckhoff TwinCAT AMS message router and database generation. The process has been aborted.

Solution:
Verify that the Beckhoff TwinCAT Automation Device Specification (ADS) Communication library is installed on the same computer as the OPC server.

Unable to generate a tag database for device <device name>. Reason: Device is not responding.

Error Type:
Warning

Possible Cause:
1. The connection between the device and the host PC is broken.
2. The named device may have been assigned an incorrect AMS Net ID address.
3. Device CPU work load is too high and has caused the process to be aborted.

Solution:
1. Verify the cabling between the PC and the PLC device.
2. Verify that the AMS Net ID address given to the named device matches that of the actual device.
3. If this error occurs frequently, decrease the tag group scan rate to reduce the work load on the PLC CPU.
4. Increase the cycle time interval under Task Configurations in TwinCAT PLC Control.

Unable to generate a tag database for device <device name>. Reason: Device returned error code <#>.

Error Type:
Warning

Possible Cause:
The cause depends on the error code(s) returned. The process has been aborted.

Solution:
The solution depends on the error code(s) returned.

See Also:
Error Codes

Unable to generate a tag database for device <device name>. Reason: Memory allocation error.

Error Type:
Warning

Possible Cause:
The memory required for database generation could not be allocated. The process was aborted.

Solution:
Close any unused applications and/or increase the amount of virtual memory. Then, try again.

Driver Error Messages

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

Unable to import from Beckhoff TwinCAT ADS Communication driver.
Unable to load Beckhoff TwinCAT ADS Communication driver.
Unable to open a communication port on the ADS router.

Unable to import from Beckhoff TwinCAT ADS Communication driver.

Error Type:

Serious

Possible Cause:

A software component that is necessary for communications with the Beckhoff TwinCAT AMS message router cannot be loaded from the TcAdsDll.dll library.

Solution:

Verify that the latest version of the Beckhoff TwinCAT Automation Device Specification (ADS) Communication Library is installed on the same computer as the OPC server. Then, restart the computer if required.

Unable to load Beckhoff TwinCAT ADS Communication driver.

Error Type:

Serious

Possible Cause:

The driver was unable to load the TcAdsDll.dll, which is necessary for communications with the Beckhoff TwinCAT AMS message router.

Solution:

Verify that the Beckhoff TwinCAT Automation Device Specification (ADS) Communication Library is installed on the same computer as the OPC server.

Unable to open a communication port on the ADS router.

Error Type:

Serious

Possible Cause:

A connection could not be established to the TwinCAT message router due to an invalid router configuration. The Beckhoff TwinCAT Automation Device Specification (ADS) Communication Library is not configured correctly.

Solution:

Follow the steps described in Setting Up an AMS Remote Connection.

Device Status Error Messages

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

Device <Device name> is not responding.

Error Type:
Warning

Result:

1. If the tag was being read, then the Read operation will not be performed and the tag will be invalidated.
2. If the tag was being written, then the Write operation for the given tag will not occur.

Possible Cause:

1. The connection between the device and the host PC is broken.
2. The named device may have been assigned an incorrect AMS Net ID address.
3. Device CPU work load is too high.
4. The response from the device took longer to receive than the amount of time specified in the "Request Timeout" device setting.

Solution:

1. Verify the cabling between the PC and the PLC device.
2. Verify that the AMS Net ID address given to the named device matches that of the actual device.
3. If this error occurs frequently, decrease the tag group scan rate to reduce the work load on the PLC CPU.
4. Increase the cycle time interval under Task Configurations in TwinCAT PLC Control.
5. Increase the Request Timeout setting so that the entire response can be handled.

Monitoring the global variable <tag address> on device <device name> to update symbol information when a change is detected.

Error Type:
Informational

Possible Cause:

The TwinCAT PLC is configured with the global variable ".SYSTEMINFO.ONLINECHANGECOUNT," which contains the number of online changes that have been made since the last complete download. The driver will monitor this variable along with all item handles to determine whether it is necessary to refresh the symbol information.

Solution:
N/A.

Unable to gather runtime information for device <device name>. Reason: <reason>.

Error Type:
Serious

Possible Cause:
The cause depends on the error reason returned.

Solution:
The solution depends on the error reason returned.

See Also:
Error Codes

Unable to perform Beckhoff compatibility on device <device name> due to memory allocation error.

Error Type:
Warning

Possible Cause:
The memory required for device tags to meet Beckhoff compatibility could not be allocated.

Solution:
Close any unused applications and/or increase the amount of virtual memory. Then, restart the server and try again.

Unable to synchronize with configuration file <file name> on device <device name> due to memory allocation error.

Error Type:
Warning

Possible Cause:
The memory required to synchronize with .tpy Runtime file could not be allocated. The original uploaded file will continue to be used.

Solution:
Close any unused applications and/or increase the amount of virtual memory. Then, restart the server and try again.

Unable to read tag <tag address> on device <device name>. Requesting symbol information for each transaction.

Error Type:
Informational

Possible Cause:
The TwinCAT PLC is not configured with the global variable ".SYSTEMINFO.ONLINECHANGECOUNT," which contains the number of online changes that have been made since the last complete download. When this variable does not exist in the PLC, the driver will request symbol information for each transaction.

Solution:
To reduce the symbol request transactions, add the ".SYSTEMINFO.ONLINECHANGECOUNT" global variable to the TwinCAT PLC project. The driver will then be able to monitor the global variable and will only request symbol information when its value changes (or when an item has an invalid handle).

See Also:
Adding SYSTEMINFO Global Variables to a TwinCAT PLC Project

Read Error Messages

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

Unable to read tag <address> on device <device name>. Actual data type is not compatible with tag of type <data type>.
Unable to read tag <address> on device <device name>. Address bounds exceeded.
Unable to read tag <address> on device <device name>. Beckhoff TwinCAT ADS DLL necessary for runtime is not loaded.
Unable to read tag <address> on device <device name>. Error Code <#>.
Unable to read tag <address> on device <device name>. Memory allocation error.
Unable to read tag <address> on device <device name>. Runtime file is not valid.
Unable to read tag <address> on device <device name>. Symbol not found in file.
Unable to read tag <address> on device <device name>. Tag data size of <size> bytes(s) exceeds actual data size of <size> byte(s).
Unable to read tag <address> on device <device name>. Tag does not meet filtering requirements.

Unable to read tag <address> on device <device name>. Actual data type is not compatible with tag of type <data type>.

Error Type:
Warning

Possible Cause:

1. A read request for tag <address> is unable to be cast.
2. A read request for tag <address> corresponds to an unsupported data type.

Solution:

1. Change the tag's data type to one that is supported.
2. Contact Technical Support so that support can be added for this type.

Note:
The data type of a item in the server should match the data type for the controller address that it references.

Unable to read tag <address> on device <device name>. Address bounds exceeded.

Error Type:
Warning

Possible Cause:
The controller's tag size is smaller than the requested data size.

Solution:
Ensure that the requested data size matches the address size in the controller.

Unable to read tag <address> on device <device name>. Beckhoff TwinCAT ADS DLL necessary for runtime is not loaded.

Error Type:
Serious

Possible Cause:
The driver was unable to load the TcAdsDll.dll, which is necessary for communications with the Beckhoff TwinCAT AMS message router.

Solution:
Verify that the Beckhoff TwinCAT Automation Device Specification (ADS) Communication Library is installed on the same computer as the OPC server.

Unable to read tag <address> on device <device name>. Error Code <#>.

Error Type:
Warning

Possible Cause:

1. The cause depends on the error code(s) returned.
2. TwinCAT does not support a read command for error code 1793 and TwinCAT version 2.10 (Build 909) or older.

Solution:

1. The solution depends on the error code(s) returned.
2. For error code 1793 and TwinCAT version 2.10 (Build 909) or older, users should update TwinCAT to the latest version. For instructions on how to do so, contact Beckhoff.

See Also:
Error Codes

Unable to read tag <address> on device <device name>. Memory allocation error.

Error Type:
Warning

Possible Cause:
The memory required for the read request could not be allocated.

Solution:
Close any unused applications and/or increase the amount of virtual memory. Then, restart the server and try again.

Unable to read tag <address> on device <device name>. Runtime file is not valid.

Error Type:
Warning

Possible Cause:
1. There is no file loaded into the project.
2. The file loaded does not have a valid *.tpy extension.
3. The file has been modified outside of TwinCAT PLC Control.

Solution:
1. Ensure that there is a valid .tpy file loaded into the project.
2. In TwinCAT PLC Control, rebuild the project and load the new .tpy file into the project.

Unable to read tag <address> on device <device name>. Symbol not found in file.

Error Type:
Warning

Possible Cause:
The tag <address> cannot be found in the .tpy file.

Solution:
1. Ensure that the correct .tpy file is loaded into the project .
2. Ensure that the correct tag <address> is entered.

Unable to read tag <address> on device <device name>. Tag data size of <size> bytes(s) exceeds actual data size of <size> byte(s).

Error Type:
Warning

Possible Cause:
A read request for the specified tag failed because the controller's tag data type is smaller than the requested data type.

Solution:
Change the tag's data type size to a size that is smaller than or equal to the ones available on the controller.

Unable to read tag <address> on device <device name>. Tag does not meet filtering requirements.

Error Type:
Warning

Possible Cause:
A read request on tag <address> does not meet the filtering requirements defined on the controller.

1. The tag does not have filtering properties.
2. The filter string entered does not match the string defined on the controller.

Solution:
Ensure that the controller is filtering the tag. Then, correct the filter string spelling.

Write Error Messages

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

Unable to write to tag <address> on device <device name>. Actual data type is not compatible with tag of type <data type>.
Unable to write to tag <address> on device <device name>. Address bounds exceeded.
Unable to write to tag <address> on device <device name>. Beckhoff TwinCAT ADS DLL necessary for runtime is not loaded.
Unable to write to tag <address> on device <device name>. Error Code <#>.
Unable to write to tag <address> on device <device name>. Memory allocation error.
Unable to write to tag <address> on device <device name>. Runtime file is not valid.
Unable to write to tag <address> on device <device name>. Symbol not found in file.
Unable to write to tag <address> on device <device name>. Tag access is read only.
Unable to write to tag <address> on device <device name>. Tag data size of <size> bytes(s) exceeds actual data size of <size> byte(s).
Unable to write to tag <address> on device <device name>. Tag does not meet filtering requirements.

Unable to write to tag <address> on device <device name>. Actual data type is not compatible with tag of type <data type>.

Error Type:
Warning

Possible Cause:
1. A write request for tag <address> is unable to be cast.
2. A write request for tag <address> corresponds to an unsupported data type.

Solution:
1. Change the tag's data type to one that is supported.
2. Contact Technical Support so that support can be added for this type.

Note:
The data type of a item in the server should match the data type for the controller address that it references.

Unable to write to tag <address> on device <device name>. Address bounds exceeded.

Error Type:
Warning

Possible Cause:
The controller's tag size is smaller than the requested data size.

Solution:
Ensure that the requested data size matches the address size in the controller.

Unable to write to tag <address> on device <device name>. Beckhoff TwinCAT ADS DLL necessary for runtime is not loaded.

Error Type:
Serious

Possible Cause:
The driver was unable to load the TcAdsDll.dll, which is necessary for communications with the Beckhoff TwinCAT AMS message router.

Solution:
Verify that the Beckhoff TwinCAT Automation Device Specification (ADS) Communication Library is installed on the same computer as the OPC server.

Unable to write to tag <address> on device <device name>. Error Code <#>.

Error Type:
Warning

Possible Cause:

1. The cause depends on the error code(s) returned.
2. TwinCAT does not support a write command for error code 1793 and TwinCAT version 2.10 (Build 909) or older.

Solution:

1. The solution depends on the error code(s) returned.
2. For error code 1793 and TwinCAT version 2.10 (Build 909) or older, users should update TwinCAT to the latest version. For instructions on how to do so, contact Beckhoff.

See Also:

Error Codes

Unable to write to tag <address> on device <device name>. Memory allocation error.

Error Type:

Warning

Possible Cause:

The memory required for the write request could not be allocated.

Solution:

Close any unused applications and/or increase the amount of virtual memory. Then, restart the server and try again.

Unable to write to tag <address> on device <device name>. Runtime file is not valid.

Error Type:

Warning

Possible Cause:

1. There is no file loaded into the project.
2. The file loaded does not have a valid *.tpy extension.
3. The file has been modified outside of TwinCAT PLC Control.

Solution:

1. Ensure that there is a valid .tpy file loaded into the project.
2. In TwinCAT PLC Control, rebuild the project and load the new .tpy file into the project.

Unable to write to tag <address> on device <device name>. Symbol not found in file.

Error Type:

Warning

Possible Cause:
The tag <address> cannot be found in the .tpy file.

Solution:
1. Ensure that the correct .tpy file is loaded into the project .
2. Ensure that the correct tag <address> is entered.

Unable to write to tag <address> on device <device name>. Tag access is Read Only

Error Type:
Warning

Possible Cause:
The write request on tag <address> does not meet OPC properties as defined on the controller.

Solution:
Change the tag properties defined on the controller to Read/Write access.

Unable to write to tag <address> on device <device name>. Tag data size of <size> bytes(s) exceeds actual data size of <size> byte(s).

Error Type:
Warning

Possible Cause:
A write request for the specified tag failed because the controller's tag data type is smaller than the requested data type.

Solution:
Change the tag's data type size to a size that is smaller than or equal to the ones available on the controller.

Unable to write to tag <address> on device <device name>. Tag does not meet filtering requirements.

Error Type:
Warning

Possible Cause:
A write request on tag <address> does not meet the filtering requirements defined on the controller.

1. The tag does not have filtering properties.
2. The filter string entered does not match the string defined on the controller.

Solution:
Ensure that the controller is filtering the tag. Then, correct the filter string spelling.

Technical Notes

For more information on specific aspects of the Beckhoff TwinCAT PLC, refer to the Technical Notes listed below.

TwinCAT Memory Warning
This notice discusses downloading a project while symbol handles are declared statically.

TwinCAT Time Slice Notes
This notice discusses the TwinCAT PLC's cyclic interval time.

TwinCAT Memory Warning

Users should note the effects that declaring symbol handles statically and using Automatic Tag Database Generation have on the Beckhoff TwinCAT project. If the project is too large, the router will not have enough memory to perform this operation. To resolve this issue, refer to and use one of the solutions described below.

Note: The router memory size is set to 2048 KB by default. The size can be increased to a maximum of 32768 KB.

Solution 1: Change the Handles from Static to Dynamic

1. Open the **TwinCAT PLC application**.
2. Click on the **Resources** tab.
3. Double-click on **Workspace**.
4. In the new window, select **TwinCAT**.
5. The window on the right should now contain a **Symbol Download** box. Select **Dynamic Symbols**.

Solution 2: Change the Router Memory Size

1. Open **TwinCAT System Manager**.
2. Expand **SYSTEM-Configuration**. Then, click on **Real-Time Settings**.
3. Click on the **Settings** tab.
4. Increase the **Router Memory** size.

Solution 3: Upload the Project Offline from a .tpy File

TwinCAT Time Slice Notes

For proper operation, there must be enough time allocated between cyclic intervals. If the time interval is too small, performance will diminish significantly. Negative effects include the following:

- More time needed to perform online Automatic Tag Database Generation.
- Slower update rates in the OPC Client.

Increasing the Cyclic Interval Time

Follow the instructions below for information on increasing the cyclic interval time.

1. Open the **TwinCAT PLC Control**.
2. Click on the **Resources** tab.
3. Double-click on **Task Configuration**, and then expand **Task Configuration** at the top of the middle window.
4. Select **Standard**.
5. In the **Task Attributes** tab, change the **Interval** value within the **Properties** box.

Appendix: SYSTEMINFO Global Variables in a TwinCAT PLC Project

Because the PLC system global variable "onlineChangeCount" is disabled by default, the PLC project must be changed to include it. The variable is added through the correct PLC library, which depends on both the version of TwinCAT PLC Control and the target PLC system type (such as PC controller, BC controller, BX controller, and so forth) being used.

Including the "onlineChangeCount" variable in a PLC will allow the server project to access PLC variables using tag handles. Without it, the server must use fully qualified Symbolic information strings in each tag read. The number of bytes in a Symbolic information string can be several times larger than the number of bytes in a tag handle. For more information on adding the "onlineChangeCount" variable, refer to the instructions below.

Note: The following example applies to TwinCAT development environment version 2.11, and describes how to include the SYSTEMINFO variables in a TwinCAT PLC project where the target system is a PC controller. It assumes there is a pre-existing PLC project. The Tc2_System library is now part of the standard TwinCAT 3 PLC template; however, the library must be added if it is not already listed in the TwinCAT 3 PLC project's Library Manager to prevent using fully qualified symbolic information strings each time tag data is accessed.

1. In **TwinCAT PLC Control**, load the PLC project. Do not login.
2. Next, locate and open the **Resources** tab at the bottom of the **Object Organizer**. Then, expand the **Resources** tree and double-click on **Library Manager**.

   Note: The Library Manager should be displayed in the TwinCAT PLC Control's work area. All libraries that are attached to a PLC project will be listed in the top left of the Library Manager. New projects should only have the "STANDARD.LIB" library attached.
3. To add another library, right-click in the open area beneath "STANDARD.LIB" and select **Additional Library**. Then, locate and double-click on **PlcSystem.lib**.

   Note: The top left of the Library Manager should now display "PlcSystem.lib".
4. Return to the **Library Manager**. Then, select **PlcSystem.lib** and open the **Data Types** tab.
5. Next, open the **SystemDataTypes** folder and select the **SYSTEMINFOTYPE** structure. The variable "onlineChangeCount" should be included within the Library Manager.
6. Login to the PLC project.
7. In the **Resources** tree, open the **Global Variables** folder. Then, double-click on the **Global_Variables** icon. At this point, the variable "onlineChangeCount" should appear as an element of structure "SystemInfo" in the work area.

   Important: The "SystemInfo" structure will only be displayed within "Global_Variables" after the user has logged in to the TwinCAT PLC Control.

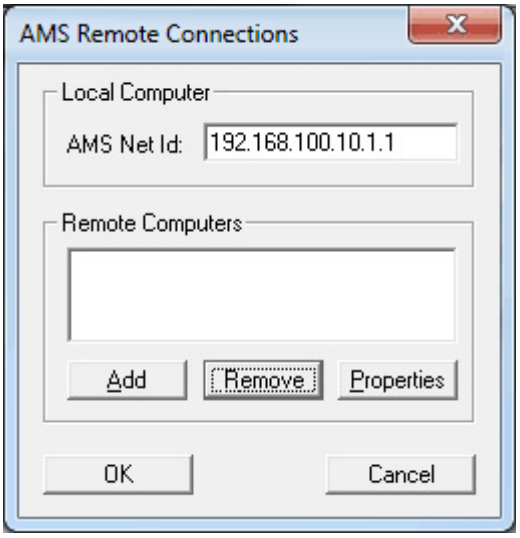
Appendix: Setting Up an AMS Remote Connection

Important: On Windows Vista and above, User Account Control (UAC) must be turned off before a remote connection may be configured with the Beckhoff TwinCAT Remote Manager Utility. If the remote connection is created before UAC is disabled, the configuration settings will be created in an incorrect location (resulting in poor performance).

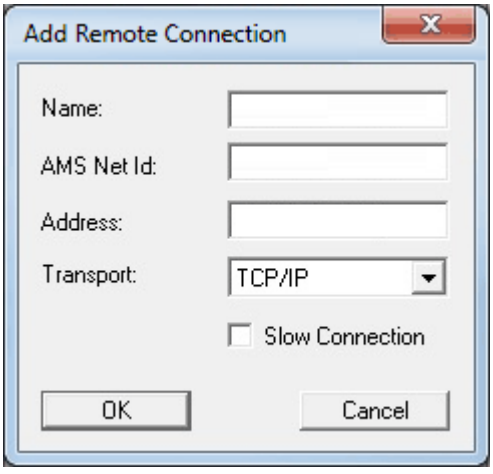
Configuration Without TwinCAT PLC Installed

A TwinCAT-PC's ADS-AMS Net ID is set using TwinCAT's Remote Manager Utility.

1. Install the **Beckhoff TwinCAT Automation Device Specification (ADS) Communication Library**, which is available online from Beckhoff.
2. Next, navigate to *TwinCAT\AdsApi\TcAdsDll*. Then, double-click on **TcAmsRemoteMgr.exe**.
3. Beckhoff TwinCAT constructs the AMS Net ID from the TCP/IP address of the local PC by default, and uses an extension of ".1.1". The AMS Net ID can, however, be chosen freely. In this example, the PC that is running the OPC server has an IP address of "192.168.100.10." The Beckhoff device has an IP address of "192.168.100.20".

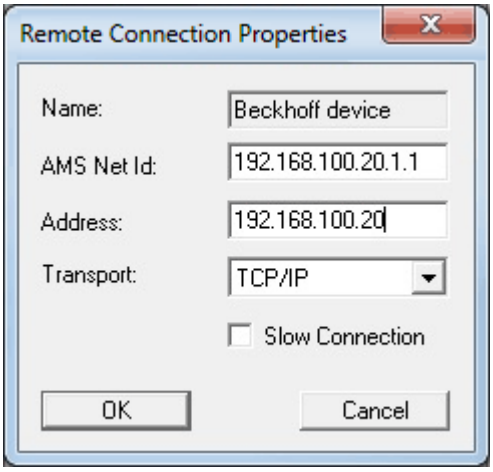


4. Next, press **Add** to include a remote computer in the connection.

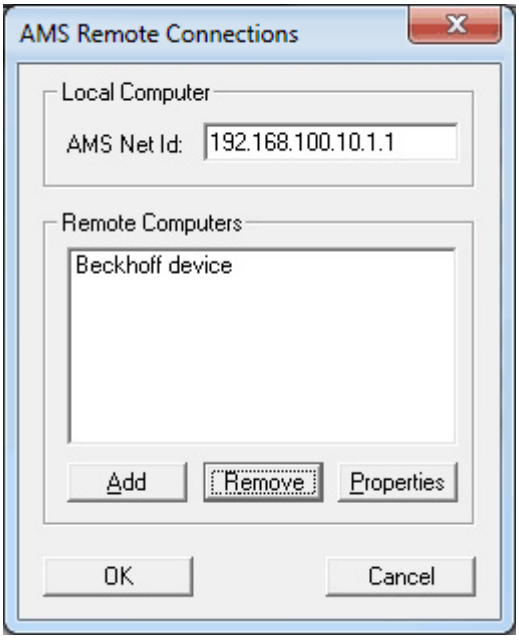


5. Then, specify the parameters.

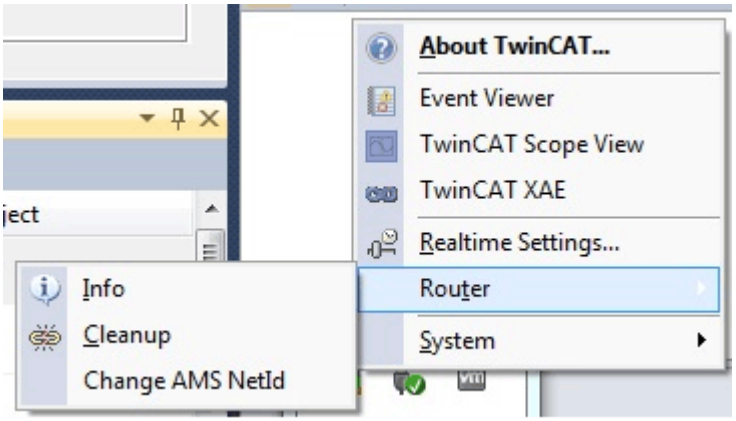
Note: This procedure must be repeated for all remote TwinCAT PLCs. The host computer must be added in the same way as the remote computers on each of the remote computers.



6. Once all the settings have been entered, press **OK**.



7. Next, click on the **TwinCAT System Service** icon located in the toolbar's notification area. Then, select **Router** | **Change AMS NetID**.

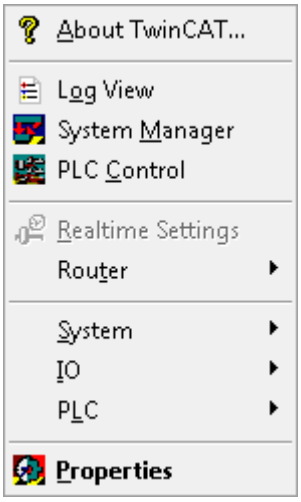


8. Verify that the **Local Computer AMS Net ID** listed matches the one defined above. If the configuration has changed, restart the computer.

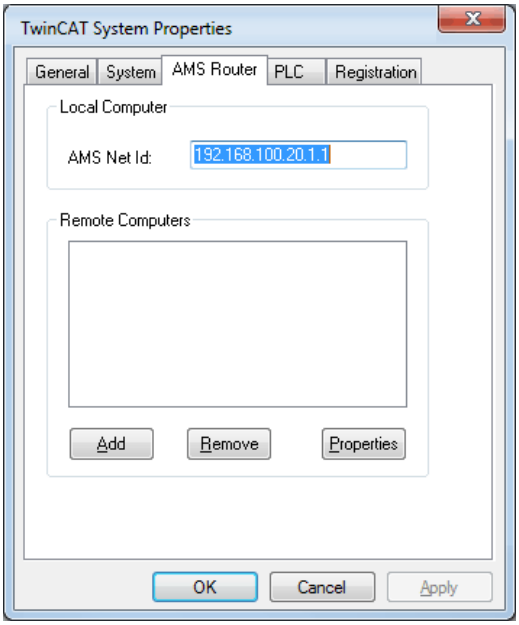
Configuration With TwinCAT 2 PLC Installed

A TwinCAT-PC's ADS-AMS Net ID is set in the TwinCAT system service. In this example, the PC that is running the OPC server has an IP address of "192.168.100.10". The Beckhoff device has an IP address of "192.168.100.20".

1. To start, click on the TwinCAT system service icon located within the toolbar's notification area. Then, select **Properties**.



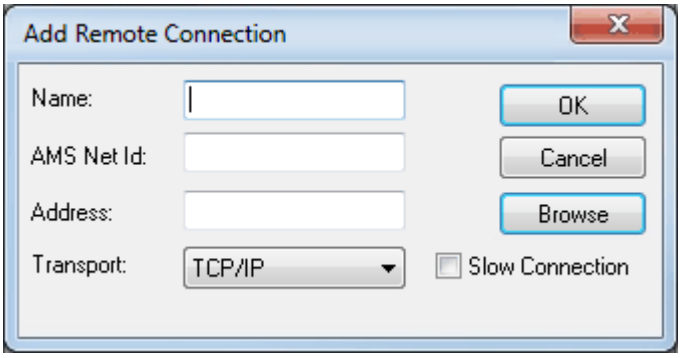
2. In **TwinCAT System Properties**, select the **AMS Router** tab.



3. In **AMS Net ID**, read or modify the desired AMS Net Identifier. The TwinCAT installation constructs the AMS Net ID from the TCP/IP address of the local PC by default, and uses an extension of .1.1 (as if it were a sub-net mask for field buses, target bus controllers, and so forth). The AMS Net ID can, however, be freely chosen.

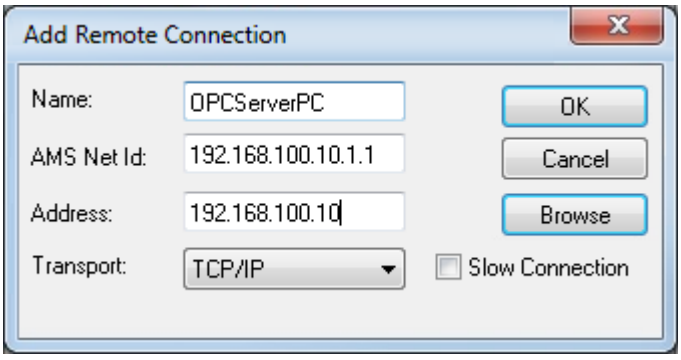
Note: When an ADS device's services are called on in the network, its AMS Net ID must be known. By inserting the target PC, TwinCAT can establish the connection between the TCP/IP address of the target PC and the AMS Net ID of the target message router address.

4. The remote computers must be known to each other to create an AMS connection between multiple computers. To do so, click **TwinCAT** | **Properties** | **AMS Router**. The AMS Net ID is composed of the TCP/IP of the local computer plus the suffix .1.1. The AMS Net ID is based on the TCP/IP address, but the relationship is not entirely fixed.
5. Next, press **Add** to include a remote computer in the connection.

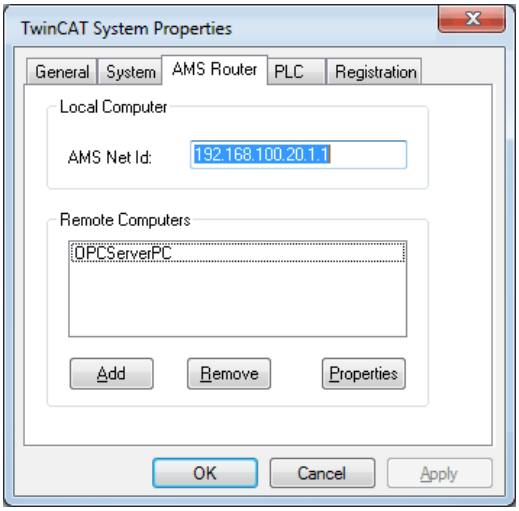


6. Then, either specify the parameters manually or select by clicking **Browse**.

Note: Each computer participating in communications must have an entry in the AMS Router Utility's Remote Computers section for every other computer that is also participating in communications.



7. Once all the settings have been entered, press **OK**.



8. Restart the computers.

Configuration With TwinCAT 3 PLC Installed

Follow the instructions listed above in **Configuration Without TwinCAT PLC Installed**, starting at step two.

Note: For more information about the AMS messaging protocol, refer to Beckhoff's documentation.

Index

A

C

D

E

G

H

I

M

N

O

P

R

S

T

U

W