Allen-Bradley ControlLogix Slave Ethernet Driver Help

Table of Contents

Allen-Bradley ControlLogix Slave Ethernet Driver Help

Help version 1.006

CONTENTS

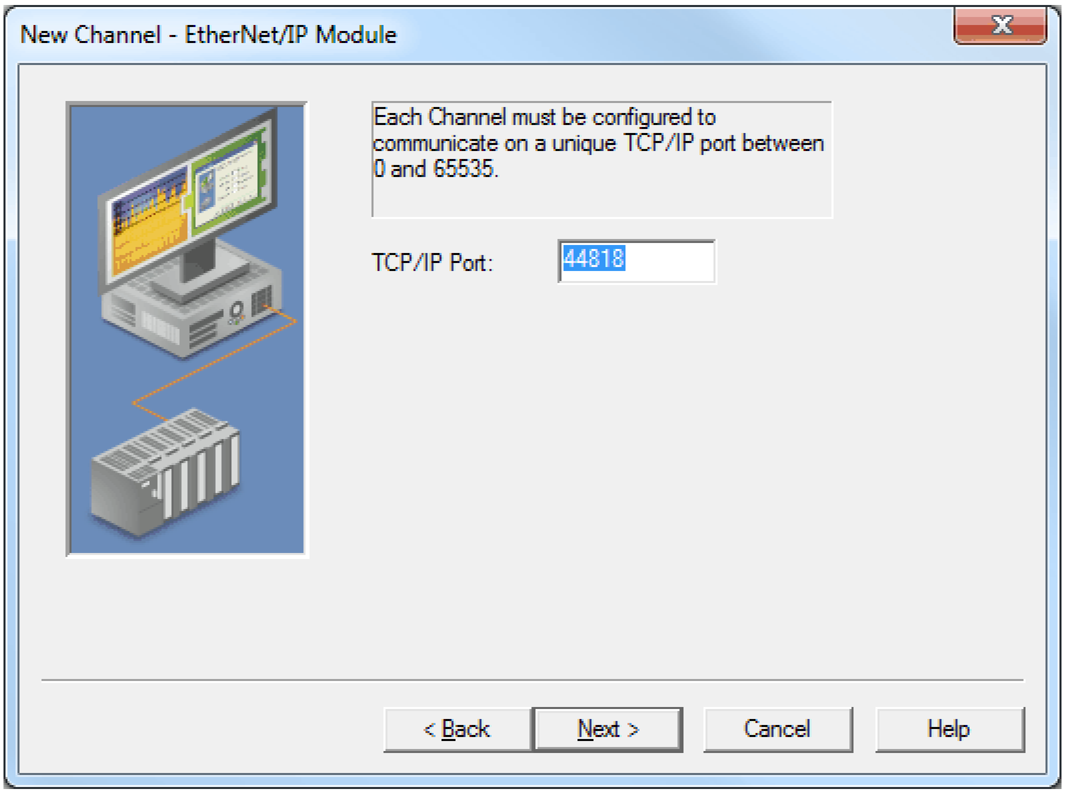
Overview

The Allen-Bradley ControlLogix Slave Ethernet Driver provides an easy and reliable way to connect Allen-Bradley ControlLogix PLCs to OPC client applications. It simulates a ControlLogix 5000 series rack containing a single EtherNet/IP module and up to 16 ControlLogix CPUs. ControlLogix 5000 series PLCs can be configured to perform CIP Data Table Read/Writes to the driver with the MSG Ladder Instruction.

Note: For more information on configuring the ControlLogix 5000 series PLC to communicate with the driver, refer to Rockwell/Allen Bradley documentation.

Channel Setup

EtherNet/IP Module


New Channel - EtherNet/IP Module

Each Channel must be configured to communicate on a unique TCP/IP port between 0 and 65535.

TCP/IP Port: 44818

Description of the parameter is as follows:

- **TCP/IP Port:** This parameter specifies the TCP/IP and UDP port that will provide a unique communication channel to the EtherNet/IP module. The valid range is 1 to 65535. The default setting is 44818.

Note: The Allen-Bradley ControlLogix Slave Ethernet Driver currently limits the number of channels to one. If the network adapter and port conflicts with another application on the host machine, the driver will fail to accept inbound EtherNet/IP connections. For more information, refer to **Error Descriptions**.

Device Setup

The Allen-Bradley ControlLogix Slave Ethernet Driver acts as a simulated ControlLogix 5000 series rack that contains a single Ethernet/IP module. The rack can contain up to sixteen ControlLogix CPUs, with one built into the EtherNet/IP module (considered local) and up to fifteen individual CPU modules (considered remote to the EtherNet/IP module). Up to 256 devices may connect to the simulated EtherNet/IP module at any time.

Supported Devices

All ControlLogix 5000 Series PLCs that support CIP Data Table Read/Write MSG Instructions and run Firmware revision 16 or higher. Both Connected and Unconnected CIP Data Table Reads or Writes are supported.

Communication Protocol

EtherNet/IP

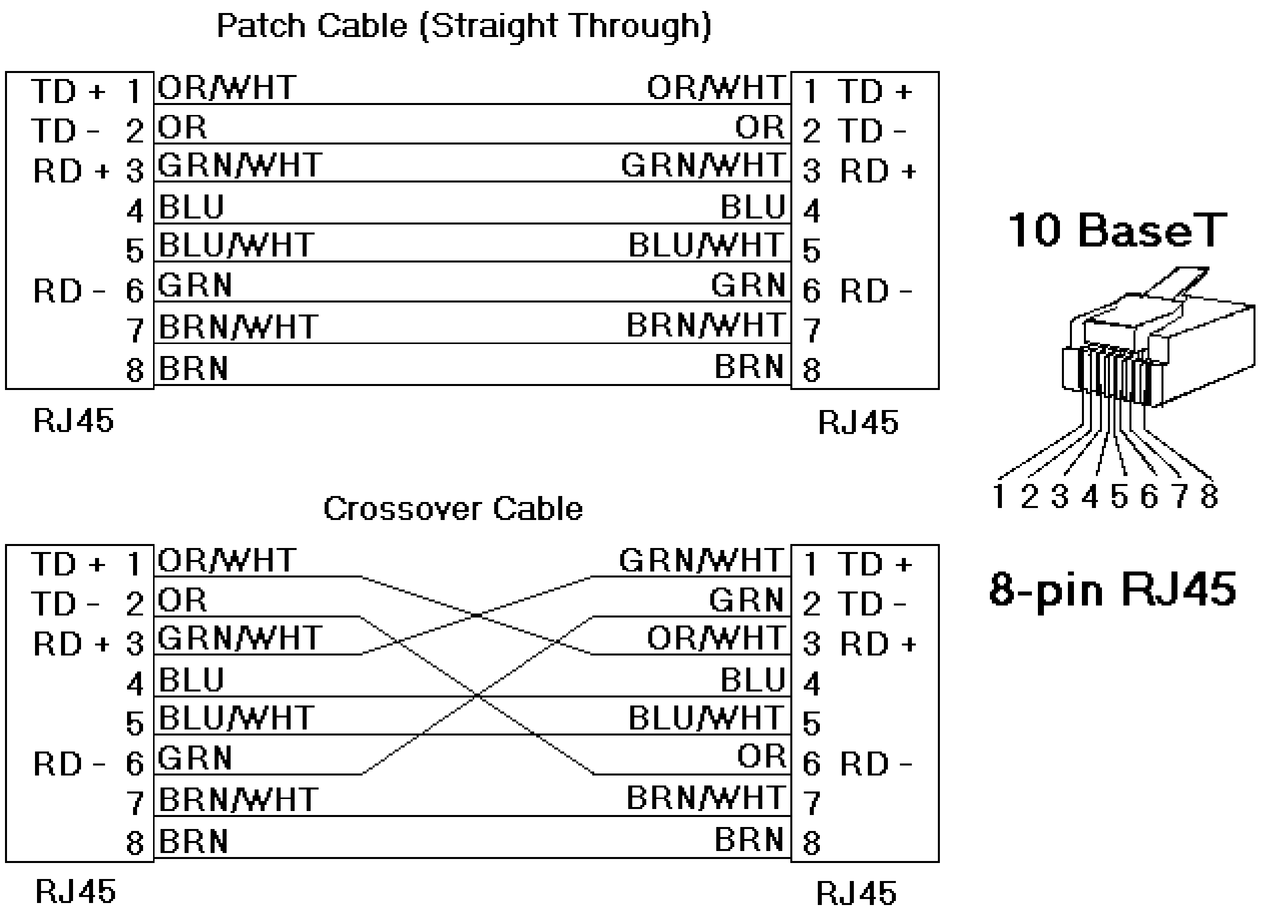
PLC Configuration

Devices on the network must be programmed to use the CIP Data Table Read/Write MSG Instruction to the driver, as well as to handle returned data. For more information on configuring the MSG Instruction, refer to Allen-Bradley's *Programming Messages In a ControlLogix System*.

Sockets

Up to 256 incoming connections are serviced simultaneously. The connections will remain open until closed by the originator.

Cable Diagrams



Master Device Configuration

Allen-Bradley ControlLogix PLCs must be programmed to issue CIP Data Table Read/Write messages to this driver using the MSG Ladder Instruction. A routing path representing the driver's configuration should be used that includes the IP Address, slot number, and optional port. For more information on the MSG Ladder Instruction, refer to the Rockwell/Allen-Bradley PLC programming documentation. The routing path that is associated with a particular slave device is provided in the Controller Module tab of Device Properties. For more information, refer to **Device Setup**.

Supported Services
Unfragmented Read
Fragmented Read
Unfragmented Write
Fragmented Write
Read/Modify/Write

Note: The ControlLogix MSG Ladder Instruction automatically decides whether to use Fragmented or Unfragmented services based on the size of the request. This is not a user configurable option.

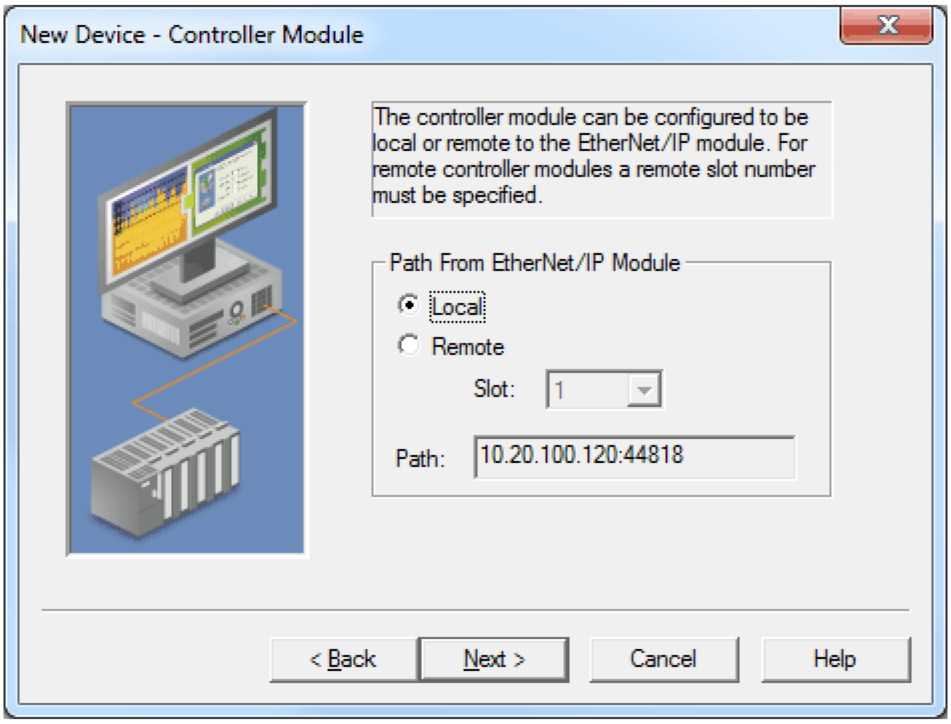
Supported Logix Types
BOOL
DWORD (BOOL array)
SINT
INT
DINT
LINT
REAL

Error Codes
The Allen-Bradley ControlLogix Slave Ethernet Driver responds to all properly formatted messages that it receives. If it cannot complete the request, a response message will be returned with a non-zero error status and an optional extended error status in the ERR and EXERR tags of the MESSAGE structure. Ladder programs should be written to handle these errors. For more information on the error codes that may be returned to master devices, refer to **Error Codes**.

Note: This driver supports CIP Data Table Read/Writes for the Logix Atomic Types list above. Although structured types are not supported, the MSG Ladder Instruction can be used to Read/Write to individual Logix Atomic Types within a structured type. For example, if the tag "MyString @ STRING" must be written to the driver, a CIP Data Table Read must be performed for "MyString.DATA" and "MyString.LEN" separately.

Controller Module
The Controller Module dialog is used to configure the device as either a Local or Remote CPU. There can be one Local CPU (which is part of the simulated EtherNet/IP Module) and up to fifteen Remote CPUs (which require a slot number for EtherNet/IP routing).

Note: Each slave device must be configured to represent a ControlLogix 5000 Series controller.


New Device - Controller Module

The controller module can be configured to be local or remote to the EtherNet/IP module. For remote controller modules a remote slot number must be specified.

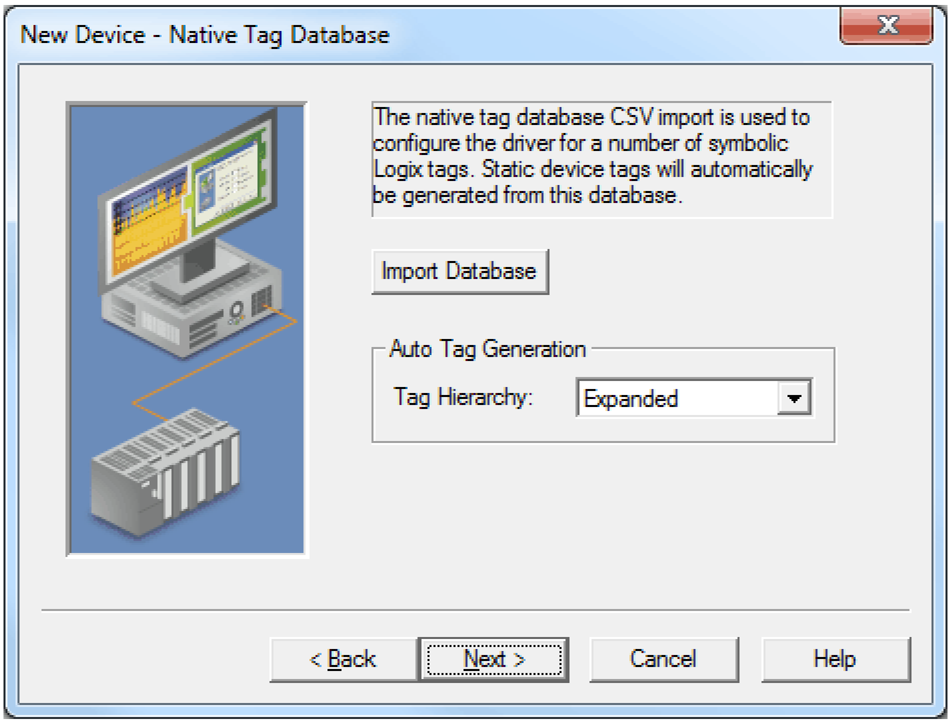
Path From EtherNet/IP Module
[✓ Local] Remote
Slot: 1
Path: 10.20.100.120:44818

Descriptions of the parameters are as follows:

- **Local:** When selected, the Controller Module will be treated as a CPU local to the simulated EtherNet/IP Module. There can only be one Local Controller Module per channel. The default setting is checked.
- **Remote:** When selected, the Controller Module will be treated as a CPU remote to the simulated EtherNet/IP Module. There can be up to fifteen remote controller modules per channel. When this option is enabled, a slot must also be specified. The default setting is unchecked.
- **Slot:** This parameter is part of the routing path to the Controller Module. It only contains slots that are currently available for the channel/device being configured. When a new slot is specified, the previous slot will be made available for use in another device.
- **Path:** This Read Only parameter represents the routing path to the Controller Module from the perspective of the Master device. It should be used during Master Device Configuration in the PLC. For more information on configuring master devices, refer to **Master Device Configuration**.

Native Tag Database

The Native Tag Database dialog is used to configure the data that the simulated Controller Module will represent, in addition to automatically generating server tags. The database will be imported as a CSV file. For more information, refer to **Native Tag Database CSV Import**.


New Device - Native Tag Database

The native tag database CSV import is used to configure the driver for a number of symbolic Logix tags. Static device tags will automatically be generated from this database.

Auto Tag Generation

Tag Hierarchy: Expanded

Descriptions of the parameters are as follows:

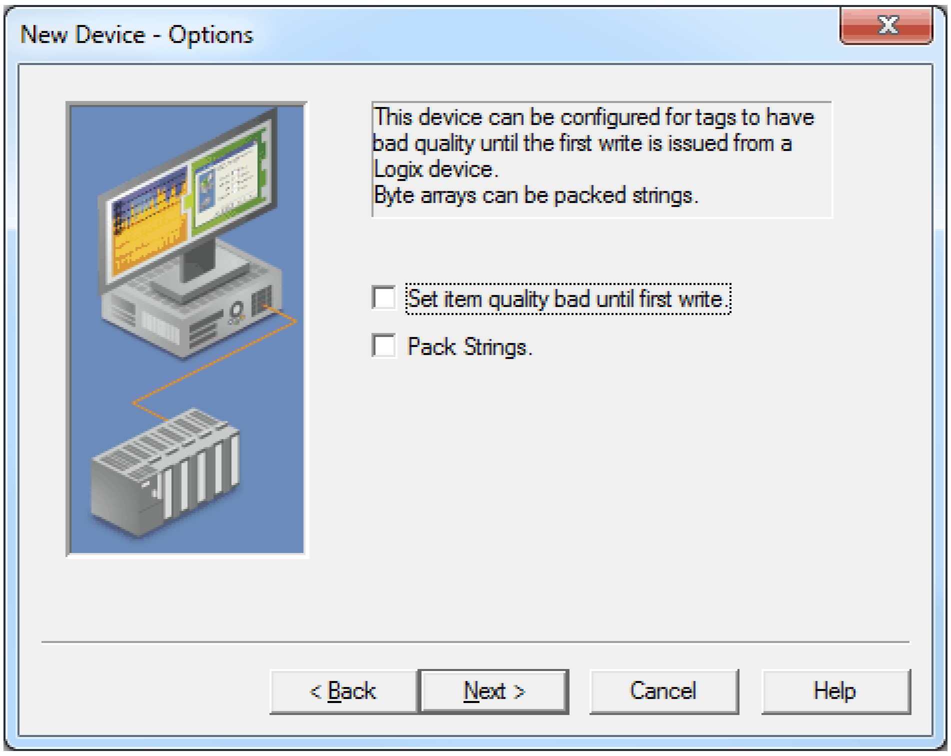
- **Import Database:** When selected, this button will invoke the Native Tag Database Import dialog for locating a CSV file to use for Native Tag Database import. Native tags cannot be imported when there are active client connections to the server.

  Note: Once the Native Tag Database is imported, the server project file will be used to maintain it. The CSV file is not needed for remote deployment. For more information, refer to **Automatic Tag Database Generation**.
- **Tag Hierarchy:** This parameter specifies the tag hierarchy. Options include Expanded and Condensed. The default setting is Expanded. Descriptions of the options are as follows:
  - **Expanded:** When selected, automatically generated client tags will be grouped similarly to RSLogix. Groups will be created for each segment following a period in the tag address, as well as structures, substructures, and arrays.
  - **Condensed:** When selected, automatically generated client tags will be grouped similarly to the tag's address. Groups will be created for each segment preceding a period.

Note: For more information on the Native Tag Database CSV format required for import, refer to **Native Tag Database CSV Import**.

Options

At startup, the Allen-Bradley ControlLogix Slave Ethernet Driver initializes integer/numeric data type values to zero (0) and strings to empty. Clients receive initial updates with Good quality by default; however, this behavior can be modified for each device.


New Device - Options

This device can be configured for tags to have bad quality until the first write is issued from a Logix device.
Byte arrays can be packed strings.

☐ Set item quality bad until first write.

☐ Pack Strings.

Descriptions of the parameters are as follows:

- **Set item quality bad until first write:** When checked, this option forces the driver to return Bad quality until a write occurs to the Native Tag. The write may occur from a client interface (such as OPC) or from a Master Device (such as a ControlLogix 5000 Series PLC). When a write occurs to a single item of an array, the entire array will be initialized and Good quality will be returned. The default setting is unchecked.
- **Pack Strings:** When checked, this option causes the string displayed in the String Tag to include all bytes of the array elements in a packed format. When unchecked, the string displayed in the String Tag will be in an unpacked format, where only the low byte of each element will be displayed.

Automatic Tag Database Generation

The Allen-Bradley ControlLogix Slave Ethernet Driver can be configured to automatically generate a list of server tags that correspond to the Native Tag Database. Native Tag Database tags must be pre-defined Logix Atomic Types, but may also be a part of a structured type.

The driver will generate a server tag for each Atomic Tag defined in the Native Tag Database. For array types, a server tag will be defined for each element of the array. Array Tags can quickly increase the number of tags imported, as well as the number of tags available in the server. Automatically generated tags are always configured with a client access of Read/Write. For more information, refer to **Native Tag Database CSV Import**.

Importing Native Tags as SINT, INT, and DINT Arrays

Native Tags that are imported as SINT, INT, and DINT arrays also have a string tag defined that uses the number of elements of the corresponding array in the tag address. Examples are as follows:

- If a Native Tag called "MySINTarray @ SINT[100]" is imported, a Static Tag with the address "MYSINTARRAY / 100" and String data type will be generated.
- If a Native Tag called "MyINTarray @ INT[100]" is imported, a Static Tag with the address "MYINTARRAY / 100" and String data type will be generated.
- If a Native Tag called "MyDINTarray @ DINT[100]" is imported, a Static Tag with the address "MYDINTARRAY / 100" and String data type will be generated.

Note: To import RSLogix5000 pre-defined Strings, the two elements contained within the String types ("STRING.DATA" and "STRING.LEN") should be defined in the Native Tag Database CSV file before the import is performed.

Tag Hierarchy

The automatically generated server tags can follow one of two hierarchies: Expanded or Condensed. The default setting is Expanded Mode.

Expanded Mode

In Expanded Mode, the automatically generated server tags follow a Group/Tag hierarchy consistent with the tag hierarchy in RSLogix5000. Groups are created for each segment that precedes a period, and are also created in logical groupings. Groups created include the following:

- Global (Controller) Scope
- Structures and Substructures
- Arrays

Note: Groups are not created for .bit addresses.

Basic Global Tags

Basic Global Tags (or non-structure, non-array tags) are placed under the Global group. Each Structure and Array Tag will be provided with its own subgroup of the parent group. By organizing the data in this fashion, the server's Tag View mimics RSLogix5000.

Note: The name of the Structure/Array subgroup also provides a description. For example, an array "tag1[1,6]" defined in the controller has a subgroup name "tag1[x,y]". In this example, *x* signifies that dimension 1 exists, and *y* signifies that dimension 2 exists. Furthermore, the tags within an array subgroup are the elements of that array unless explicitly limited. The tags within a structure subgroup are the structure members themselves. A structure that contains an array will have an array subgroup of the structure group created as well.

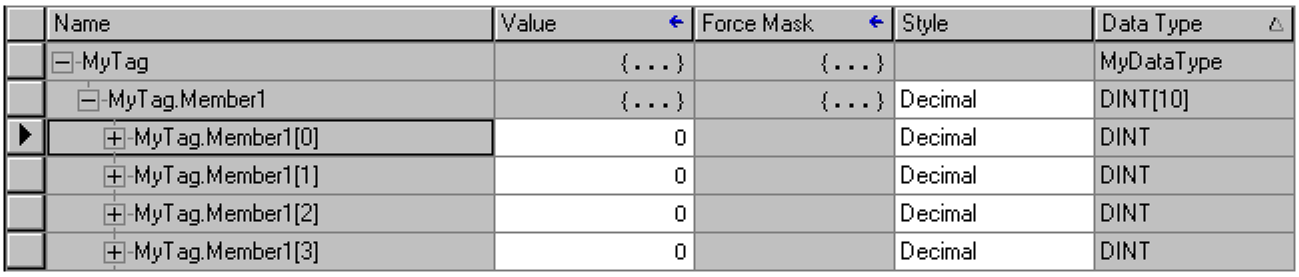
Array Tags

A group will be created for each array that contains the array's elements. Group names have the notation *<array name>[x,y,z]*, where:

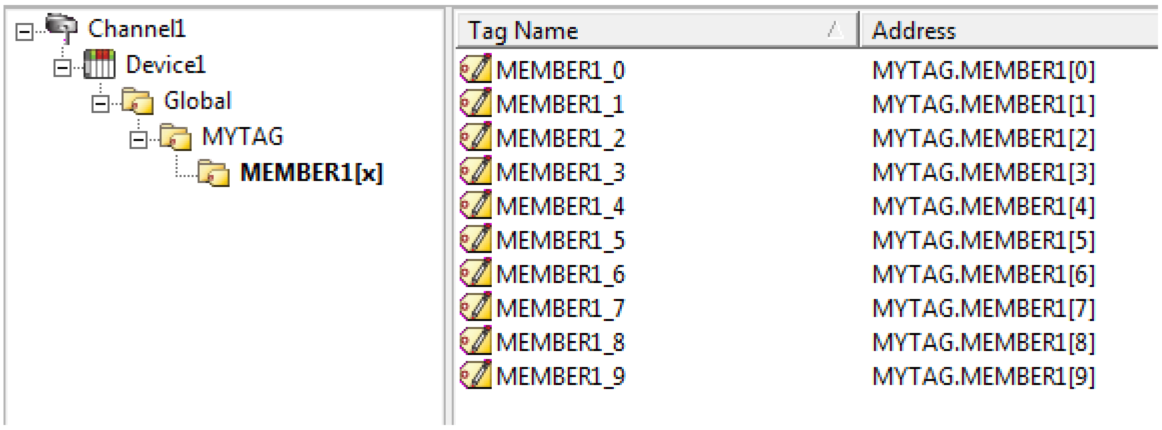
- **[x,y,z]:** 3 dimensional array.
- **[x,y]:** 2 dimensional array
- **[x]:** 1 dimensional array

Note: Array Tags have the notation *<tag element>_XXXXX_YYYYY_ZZZZZ*. For example, element "tag1 [12,2,987]" has the tag name "tag1_12_2_987".

Simple Example



Name	Value	Force Mask	Style	Data Type
MyTag	{...}	{...}		MyDataType
MyTag.Member1	{...}	{...}	Decimal	DINT[10]
MyTag.Member1[0]	0		Decimal	DINT
MyTag.Member1[1]	0		Decimal	DINT
MyTag.Member1[2]	0		Decimal	DINT
MyTag.Member1[3]	0		Decimal	DINT



Channel1
Device1
Global
MYTAG
MEMBER1[x]

Tag Name	Address
MEMBER1_0	MYTAG.MEMBER1[0]
MEMBER1_1	MYTAG.MEMBER1[1]
MEMBER1_2	MYTAG.MEMBER1[2]
MEMBER1_3	MYTAG.MEMBER1[3]
MEMBER1_4	MYTAG.MEMBER1[4]
MEMBER1_5	MYTAG.MEMBER1[5]
MEMBER1_6	MYTAG.MEMBER1[6]
MEMBER1_7	MYTAG.MEMBER1[7]
MEMBER1_8	MYTAG.MEMBER1[8]
MEMBER1_9	MYTAG.MEMBER1[9]

Complex Example
A Logix Tag defined with the address "MyStructArray[0].MySubStruct.Data" would be represented in the following groups: "Global," "MYSTRUCTARRAY[x]," "MYSTRUCTARRAY[0]," and "MYSUBSTRUCT". The tag "DATA" would be in the last group. The static reference to "DATA" would be "Channel1.Device1.Global.MYSTRUCTARRAY[X].MYSTRUCTARRAY[0].MYSUBSTRUCT.DATA". The dynamic reference would be "Channel1.Device1. MyStructArray[0].MySubStruct.Data". For more information, refer to "Static Tags (User-Defined)" and "Dynamic Tags" in server help file.

Condensed Mode

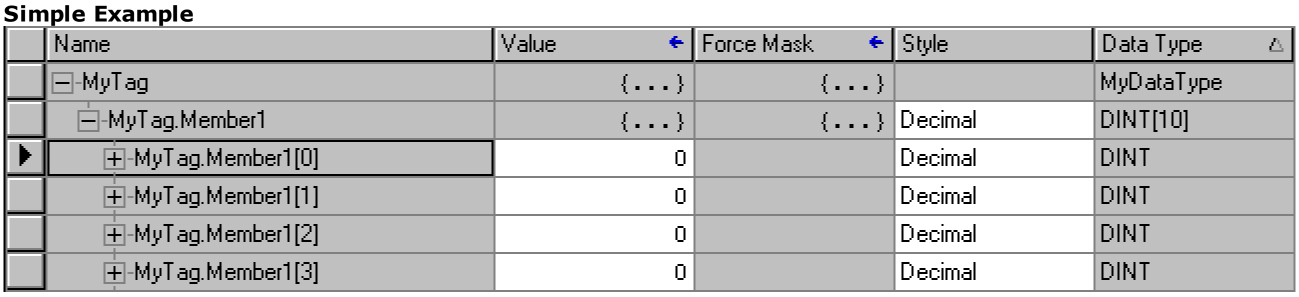
In Condensed Mode, the automatically generated server tags follow a group/tag hierarchy consistent with the tag's address. Groups will be created for each segment that precedes the period. Groups created include the following:

- Program Scope
- Structures and Substructures

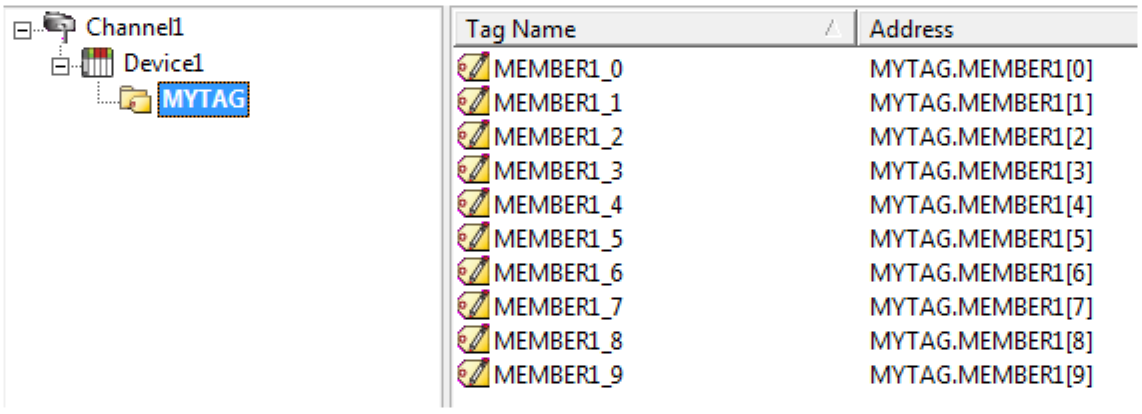
Note 1: Groups are not created for arrays or .bit addresses.

Note 2: Tag or structure member names that start with an underscore will be converted to "U". This is required because the server does not support leading underscores in tag name fields.

Simple Example



Name	Value	Force Mask	Style	Data Type
MyTag	{...}	{...}		MyDataType
MyTag.Member1	{...}	{...}	Decimal	DINT[10]
MyTag.Member1[0]	0		Decimal	DINT
MyTag.Member1[1]	0		Decimal	DINT
MyTag.Member1[2]	0		Decimal	DINT
MyTag.Member1[3]	0		Decimal	DINT



Tag Name	Address
MEMBER1_0	MYTAG.MEMBER1[0]
MEMBER1_1	MYTAG.MEMBER1[1]
MEMBER1_2	MYTAG.MEMBER1[2]
MEMBER1_3	MYTAG.MEMBER1[3]
MEMBER1_4	MYTAG.MEMBER1[4]
MEMBER1_5	MYTAG.MEMBER1[5]
MEMBER1_6	MYTAG.MEMBER1[6]
MEMBER1_7	MYTAG.MEMBER1[7]
MEMBER1_8	MYTAG.MEMBER1[8]
MEMBER1_9	MYTAG.MEMBER1[9]

Complex Example
A Logix Tag defined with address "MyStructArray[0].MySubStruct.Data" would be represented in the following groups: "MYSTRUCTARRAY[0]"and "MYSUBSTRUCT". The tag "DATA" would be in the last group. The static reference to "DATA" would be "Channel1.Device1.MYSTRUCTARRAY[0].MYSUBSTRUCT.DATA" and the dynamic reference would be "Channel1.Device1.MyStructArray[0].MySubStruct.Data".

Native Tag Database CSV Import

A CSV file specifies the Native Tags that each device can represent. It is used once for tag import, and is not required for automatic tag database generation or remote deployment. The following CSV header must be used for Native Tag import:

Logix Address	Logix DataType	External Access	Description

Note: A template Native Tag Database CSV file is included for reference in *<Server Installation Directory>/Drivers/controllogix_unsolicited_ethernet/import_template.csv*.

Logix Address

Restrictions on the Logix Address are consistent with RSLogix5000 requirements, which correspond to the following IEC 1131-3 identifier rules:

- Must begin with an alphabetic character (A-Z, a-z) or an underscore.
- Can only contain alphanumeric characters and underscores.
- Can have as many as 40 characters in each segment.
- Cannot have consecutive underscores.
- Are not case sensitive.

Tags that do not have a unique Logix Address or meet the identifier requirements above will fail to import, causing a message with the specified Logix Address to be posted to the server's Event Log.

Logix DataType

The following pre-defined Logix Atomic Types are supported:

Logix DataType	**Supported Data Types**
BOOL	Boolean
SINT	Char, Byte
INT	Short, Word
DINT	Long, DWord
LINT	Double, Date
REAL	Float

Note: Other pre-defined or user-defined complex (structured) data types are not supported. Structured data can be imported by qualifying the Logix Address down to the atomic type. For example, there exists a structured type called TIME, which is described as the following:

TIME
{
HOUR @ SINT
MIN @ SINT

SEC @ SINT
}

The structure can be broken down and the atomic member imported as "TIME.HOUR," "TIME.MIN," and "TIME.SEC" with the associated Logix DataType, External Access, and Description following the CSV format outlined above. All unsupported Logix DataType values specified in the CSV import are defaulted to DINT so that the import succeeds.

External Access

The External Access specifies the master devices' Read/Write privileges. This access does not apply to client tags, which always have a client access of Read/Write by default. The following external access types are supported: all other values specified will be set to Read/Write.

- **R/W:** Master Devices have Read/Write permissions to the Native Tag.
- **RO:** Master Devices have Read Only permissions. All write attempts fail with the appropriate error (CIP error 0x0F).

Description

Descriptions are used during automatic tag database generation, and will be truncated to 64 characters. The Description field must be present, but may be left blank.

Data Types Description

Data Types	Description
Boolean	Single bit
Char	Signed 8 bit value bit 0 is the low bit bit 6 is the high bit bit 7 is the sign bit
Byte	Unsigned 8 bit value bit 0 is the low bit bit 7 is the high bit
Short	Signed 16 bit value bit 0 is the low bit bit 14 is the high bit bit 15 is the sign bit
Word	Unsigned 16 bit value bit 0 is the low bit bit 15 is the high bit
Long	Signed 32 bit value bit 0 is the low bit bit 30 is the high bit bit 31 is the sign bit
DWord	Unsigned 32 bit value bit 0 is the low bit bit 31 is the high bit
Float	32 bit floating point value bit 0 is the low bit bit 31 is the high bit
Double	64 bit floating point value bit 0 is the low bit bit 63 is the high bit
String	Typically null terminated, null padded, or blank padded ASCII string.
Date	64 bit floating point value.

Note: For a description of Logix platform-specific data types, refer to **Address Descriptions**.

Address Descriptions

The Allen-Bradley ControlLogix Slave Ethernet Driver supports symbolic tag-based addressing.

Logix Tag-Based Addressing

This driver uses a tag or symbol-based addressing structure that is commonly referred to as Logix or Native Tags (which is consistent with Rockwell Automation's Integrated Architecture). These tags differ from conventional PLC data items in that the tag name is the address, not a physical or logical address.

The Allen-Bradley ControlLogix Slave Ethernet Driver allows users to access the controller's atomic data types BOOL, SINT, INT, DINT, LINT, and REAL. Although some of the pre-defined types are structures, they are ultimately based on these atomic data types. As such, all non-structure (atomic) members of a structure are accessible. For example, a TIMER cannot be assigned to a server tag but an atomic member of the TIMER can be (such as TIMER.EN, TIMER.ACC, and so forth). If a structure member is a structure itself, both structures must be expanded to access an atomic member of the substructure. This is more common with user and module-defined types, and is not found in any of the pre-defined types.

Atomic Data Type	Description		Range
BOOL	Single bit value	VT_BOOL	0, 1
SINT	Signed 8-bit value	VT_I1	-128 to 127
INT	Signed 16-bit value	VT_I2	-32,768 to 32,767
DINT	Signed 32-bit value	VT_I4	-2,147,483,648 to 2,147,483,647
LINT	Signed 64-bit value	VT_R8	-9.22337E18 to 9.22336E18
REAL	32-bit IEEE floating point	VT_R4	1.1755 E-38 to 3.403E38 0 -3.403E-38 to -1.1755

Client/Server Tag Address Rules

Logix Tag names correspond to Client/Server Tag addresses. Logix Tag names, which are entered via RSLogix5000, follow the IEC 1131-3 identifier rules. Client/Server Tag addresses follow these same rules. They are as follows:

- Must begin with an alphabetic (A-Z, a-z) character or an underscore.
- Can only contain alphanumeric characters and underscores.
- Can have as many as 40 characters in each segment.
- Cannot have consecutive underscores.
- Are not case sensitive.

Note 1: Tag name assignment in the server differs from address assignment in that names cannot begin with an underscore.

Note 2: In order for tags to be properly validated, a Native Tag that represents the Static Client Tag must exist in the Native Tag Database.

Address Formats

There are several ways to address a Logix Tag statically in the server or dynamically from a client. The selected format depends both on the type of tag and how it will be used. For example, when accessing a bit within a SINT-type tag, the bit format would be used. For more information on address format and syntax, refer to the table below.

Note: All formats are native to RSLogix5000 except for Array and String. When referencing an atomic data type, users can copy an RSLogix5000 tag name and then paste it into the server's Tag Address parameter: it will be valid as long as the corresponding Native Tag exists in the Native Tag Database.

Format	Syntax	Example	Notes
Standard	<Logix Tag Name>	tag_1	The tag cannot be an array.
Array Element	<Logix Array Tag Name> [dim 1, dim2, dim 3]	tag_1 [2, 58, 547] tag_1 [0, 3]	Dimension Range = 1 to 3. Element Range = 0 to

			65535.

Array w/o Offset*	<Logix Array Tag Name> {# columns}
	<Logix Array Tag Name> {# rows}{# columns}
		tag_1 {8}
		tag_1 {2}{4}
			Dimension Range = 1 to 2.
			Element Range = 1 to 65535.
			
			The number of elements to Read/Write equals the # of rows times the # of columns. If no rows are specified, the number of rows default to 1.
			
			The array begins at a zero offset (array index equals 0 for all dimension-s).

Array w/ Offset*	<Logix Array Element Tag> {# columns}
	<Logix Array Element Tag> {# rows}{# columns}
		tag_1 [2, 3] {10}
		tag_1 [2, 3] 2}{5}
			The array begins at an offset specified by the dimensions in the Array Element Tag. The array always covers the highest dimension. Thus, "tag_ 1[2,3]{10}" would produce an array of elements tag_1[2,3] -> tag_1 [2,13].

Bit	<Logix Tag Name>.bit <Logix Tag Name>.[bit]	tag_1.0 tag_1.[0]	Bit Range = 0 to 31. If the tag is an array, it must be a BOOL array; otherwise, tag cannot be an array.
String	<Logix Tag Name>.Data/<Maximum string length>	tag_1.Data/4	Length

			Range = 1 to 65535. The maximum number of characters that can Read/Write to the string.

*Because this format may request more than one element, the order in which array data is passed depends on the dimension of the Logix Array Tag. For example, if the rows multiplied by the columns is 4 and the Controller Tag is a 3X3 element array, then the elements that are being referenced are "array_tag [0,0]," "array_tag [0,1]," "array_tag [0,2]," and "array_tag [1,0]" in that order. The results would be different if the Controller Tag were a 2X10 element array.

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

Tag Scope

Global Tags

Global Tags are Logix Tags that have global scope in the controller. Any program or task can access Global Tags; however, the number of ways a Global Tag can be referenced depends on its Logix data type and the address format being used.

Program Tags

Program Tags are identical to Global Tags except that their scope is local to the program in which it is defined. The Allen-Bradley ControlLogix Slave Ethernet Driver does not currently support importing Native Tags with a program designation.

Structure Tag Addressing

Logix Structure Tags are tags with one or more member tags (which can be atomic or structured).
<structure name> . <atomic-type tag>

This implies that a substructure would be addressed as the following:
<structure name> . <substructure name> .<atomic-type tag>

Arrays of structures would be addressed as the following:
<structure array name> [dim1, dim2, dim3] . <atomic-type tag>

This implies that an array of substructures would be addressed as the following:
<structure name> . <substructure array name> [dim1, dim2, dim3] . <atomic-type tag>

Note: The examples above display a few of the addressing possibilities that involve structures, and are only provided as an introduction to structure addressing. For more information, refer to Rockwell/Allen-Bradley documentation.

Advanced Addressing

Users have several options for symbolic addressing that can be included in the Symbolic Tag address. The following restrictions have been placed on the data type for the bit and array addressing syntaxes:

- For bit syntaxes, the index cannot exceed the bit size for the data type. For example, "MyDint @ Dint" is imported as a Native Tag. The bit index cannot exceed 31 (because DINTs are 32-bit signed values).
- For array syntaxes, the offset and number of elements in the array cannot exceed the number of elements in the associated Native Tag. For example, "MyDintArray @ DINT[10]" is imported as a Native Tag. A Static Tag with addresses "MYDINTARRAY[0] {5}" and "MYDINTARRAY[4] {5}" are valid because the arrays include the first and last 5 elements of the Native Tag respectively. A Static Tag with address "MYDINTARRAY[5]{10}" is invalid because the tag is asking for 10 DINTs beginning at offset 5 and the Native Tag array is not that large.

For more information on advanced topics, refer to the table below.

Element	Syntax	Example	Notes
Standard	<tag name>	tag_1	N/A.

Array w/o Offset	<array tag name> {# of columns}
	
	<array tag name> {# of rows} {# of columns}
		tag_1 {8}
		
		tag_1 {2} {4}
			The number of elements to Read/Write equals # of rows times # of columns. If no rows are specified, # of rows defaults to 1. At least 1 element of the array must be addressed.
			
			The array begins at a zero offset (array index equals 0 for all dimension-s).

Array w/ Offset	<array element tag> [offset] {# of columns}
	
	<array element tag> [offset]{# of rows}{# of columns}
		tag_1 [5] {8}
		
		tag_1[5] {2} {4}
			The number of elements to Read/Write equals # of rows * # of columns. If no rows are specified, # of rows defaults to 1. At least 1 element of the array must be addressed.
			
			The array begins at a zero offset (array index equals 0 for all dimension-s).

Bit	<tag name> . bit <tag name> . [bit]	tag_1 . 0 tag_1 . [0]	N/A. N/A.
String	<tag name> / <element count>	tag_1 / 4	The element count must be at least 1. The number of characters to Read/Write depends on

			the Pack Strings parameter.*

*When checked, the number of characters equals the element count times the element size (4 elements of an INT array indicates 8 characters). When unchecked, the number of characters equals the element count (4 elements of an INT array indicates 4 characters). For more information, refer to **Options**.

Ordering of Logix Array Data

Since Native Tags support up to three dimensional arrays, the ordering of Logix Array Data is mapped to a 2-dimensional OPC array.

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 Codes

The Allen-Bradley ControlLogix Slave Ethernet Driver may return the following error codes. For more information on a specific type of error code, select a link from the list below.

EtherNet/IP Encapsulation Error Codes

The Allen-Bradley ControlLogix Slave Ethernet Driver may return the following error codes.

Note: The error codes are in hexadecimal.

Error	Description
0001	Command not handled.
0002	Memory not available for command.
0003	Poorly formed or incomplete data.
0064	Invalid Session ID.
0065	Invalid length in header.
0069	Requested protocol version not supported.

CIP Error Codes

The error codes are in hexadecimal.

Error	Description
01	Connection failure.*
02	Insufficient resources.
03	Parameter value invalid.
04	IOI could not be deciphered or tag does not exist.
05	Unknown destination.
06	Data requested would not fit in response packet.
08	Unsupported service.
0F	Permission denied.
13	Insufficient command data/parameter specified to execute service.
26	The number of IOI words specified does not match IOI word count.
FF	General Error.**

*__See Also:__ **0x01 Extended Error Codes**
__See Also:__ **0xFF Extended Error Codes

0x01 Extended Error Codes

The Allen-Bradley ControlLogix Slave Ethernet Driver may return the following extended errors for CIP error 0x01.

Note: The error codes are in hexadecimal.

Error	Description
0x0205	Unconnected Send parameter error.
0x0312	Link address is not available.
0x0318	Link address to self is invalid.

0xFF Extended Error Codes

The Allen-Bradley ControlLogix Slave Ethernet Driver may return the following extended errors for CIP error 0xFF.

Note: The error codes are in hexadecimal.

Error	Description

2104	Address is out of range.
2105	Attempt to access beyond the end of the data object.
2107	The data type is invalid or not supported.

Error Descriptions

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

Address Validation Error Messages

Address '<address>' is out of range for the specified device or register
Array size is out of range for address '<address>'
Array support is not available for the specified address: '<address>'
Data Type '<type>' is not valid for device address '<address>'
Device address '<address>' contains a syntax error
Memory could not be allocated for tag with address '<address>' on device '<device name>'
Missing address

Automatic Tag Database Generation Error Messages

Failed to perform auto-tag generation for device '<device>' due to low resources

Communication Error Messages

Failed to start unsolicited Logix server for channel '<channel name>'
The TCP/IP Port must be between 1 and 65535, using the default port (44818) for channel '<channel name>'

Device Specific Error Messages

Devices '<device name >' and '<device name>' are currently configured to use the same Path from EtherNet/IP Module. Each channel must have a unique Path from EtherNet/IP Module

Native Tag Database Import Error Messages

Error importing Native Tag '<address>'. Total database tag data size limited to 128 kB
Error importing Native Tag database
Error importing Native Tag database. Duplicate field name: '<field name>'
Error importing Native Tag database. File encoding not supported
Error importing Native Tag database. Incomplete tag field identification record
Error importing Native Tag database. Missing tag field identification record
Error importing Native Tag database. Unable to open file, <filename>

Error importing Native Tag database. Unrecognized field name: '<field name>'
Invalid Native Tag address: '<address>'. Individual tag size limited to 128 kB
Invalid Native Tag address: '<tag address>'. Duplicate Native Tag addresses are not allowed

Address Validation

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

Address Validation Error Messages

Address '<address>' is out of range for the specified device or register
Array size is out of range for address '<address>'
Array support is not available for the specified address: '<address>'
Data Type '<type>' is not valid for device address '<address>'
Device address '<address>' contains a syntax error
Memory could not be allocated for tag with address '<address>' on device '<device name>'
Missing address

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

Error Type:
Warning

Possible Cause:
A tag address that has been specified statically references a location that is beyond the range of the device's supported locations.

Solution:
1. Verify that the address is correct; if it is not, re-enter it in the client application.
2. Verify that the Native Tag address exists and meets the bit and/or array requirements of the Static Tag.

3. Verify that the string's element count is less than or equal to the controller's pre-defined element count for the corresponding array.

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:
1. Specify a smaller value for the array.
2. Specify a different starting point by re-entering the address in the client application.

Array support is not available for the specified address: '<address>'

Error Type:
Warning

Possible Cause:
A tag address that has been specified statically contains an array reference for an address type that does not support arrays

Solution:
1. Re-enter the address in the client application to remove the array reference.
2. Correct the address type.

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. An invalid address format was used.

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

See Also:
Address Formats

Memory could not be allocated for tag with address '<address>' on device '<device name>

Error Type:
Warning

Possible Cause:
The resources that are needed to build a tag could not be allocated. The tag was not added to the project.

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

Missing address

Error Type:
Warning

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

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

Automatic Tag Database Generation Error Messages
Failed to perform auto-tag generation for device '<device>' due to low resources

Failed to perform auto-tag generation for device '<device>' due to low resources

Error Type:
Fatal

Possible Cause:
There are not enough system resources to perform automatic tag database generation.

Solution:
Verify resource usage and then try again.

Communication Error Messages

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

Communication Error Messages
Failed to start unsolicited Logix server for channel '<channel name>'
The TCP/IP Port must be between 1 and 65535, using the default port (44818) for channel '<channel name>'

Failed to start unsolicited Logix server for channel '<channel name>'

Error Type:
Fatal

Possible Cause:
The driver was unable to bind and listen on the specified IP/Port.

Solution:
Verify that the specified IP/Port (both TCP and UDP) is not being used by another application.

The TCP/IP Port must be between 1 and 65535, using the default port (44818) for channel '<channel name>'

Error Type:
Warning

Possible Cause:
A project was loaded that specifies a TCP/IP port of 0, but this option is no longer supported.

Solution:
Use the default port (44818) or select a port within the valid range (1 to 65535).

Device Specific Error Messages

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

Device Specific Error Messages
Devices '<device name >' and '<device name>' are currently configured to use the same Path from EtherNet/IP Module. Each channel must have a unique Path from EtherNet/IP Module

Devices '<device name >' and '<device name>' are currently configured to use the same Path from EtherNet/IP Module. Each channel must have a unique Path from EtherNet/IP Module

Error Type:
Warning

Possible Cause:
The device being configured has been configured for a CPU type and slot number that is already being used.

Solution:
1. Select a different CPU type (Local or Remote).
2. Select a different slot number if the device is configured as a Remote controller module.

Native Tag Database Import Error Messages

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

Native Tag Database Import Error Messages
Error importing Native Tag '<address>'. Total database tag data size limited to 128 kB
Error importing Native Tag database
Error importing Native Tag database. Duplicate field name: '<field name>'
Error importing Native Tag database. File encoding not supported
Error importing Native Tag database. Incomplete tag field identification record
Error importing Native Tag database. Missing tag field identification record
Error importing Native Tag database. Unable to open file, <filename>

Error importing Native Tag database. Unrecognized field name: '<field name>'
Invalid Native Tag address: '<address>'. Individual tag size limited to 128 kB
Invalid Native Tag address: '<tag address>'. Duplicate Native Tag addresses are not allowed

Error importing Native Tag '<address>'. Total database tag data size limited to 128 kB

Error Type:
Warning

Possible Cause:
The CSV file being imported contains tag definitions that require more than 128 kB of memory.

Solution:
This is usually due to large arrays being defined for a Native Tag. If more than 128 kB of memory is required, create a new device and then split the Native Tag Database between multiple devices.

Error importing Native Tag database

Error Type:
Warning

Possible Cause:
An unexpected error has been encountered.

Solution:
Verify that the CSV file being imported is properly formatted.

Error importing Native Tag database. Duplicate field name: '<field name>'

Error Type:
Warning

Possible Cause:
The CSV file being imported contains multiple definitions of the same field name.

Solution:
Verify that there are no duplicated fields in the CSV file.

Note:
Supported field names include "Logix Address," "Logix Data Type," "External Access," and "Description."

Error importing Native Tag database. File encoding not supported

Error Type:
Warning

Possible Cause:
The CSV file being imported uses a file encoding that is not supported.

Solution:
Ensure that the CSV file uses ANSI or UTF-8 encodings. No other file encodings are supported.

Error importing Native Tag database. Incomplete tag field identification record

Error Type:
Warning

Possible Cause:
The CSV file being imported does not contain a header.

Solution:
Verify that there are no missing fields in the CSV file.

Note:
Supported field names include "Logix Address," "Logix DataType," "External Access," and "Description."

Error importing Native Tag database. Missing tag field identification record

Error Type:
Warning

Possible Cause:
The CSV file being imported does not contain a header.

Solution:
Verify that there are no missing fields in the CSV file.

Note:
Supported field names include "Logix Address," "Logix DataType," "External Access," and "Description."

Error importing Native Tag database. Unable to open file, <filename>

Error Type:
Warning

Possible Cause:
The CSV file being imported is being used by another application that prevents read access to the file.

Solution:
Ensure that there are no other applications using the CSV file that is being imported.

Error importing Native Tag database. Unrecognized field name: '<field name>'

Error Type:
Warning

Possible Cause:
The CSV file defines a field that is not supported by the Native Tag Database import.

Solution:
Verify that there are no unintended fields in the CSV file.

Note:
Supported field names include "Logix Address," "Logix DataType," "External Access," and "Description."

Invalid Native Tag address: '<address>'. Individual tag size limited to 128 kB

Error Type:
Warning

Possible Cause:
The CSV file being imported contains a single tag definition that requires more than 128 kB of memory to represent.

Solution:
This is usually due to large arrays being defined for a Native Tag. Native tags cannot be defined that require more than 128 kB of memory.

Invalid Native Tag address: '<tag address>'. Duplicate Native Tag addresses are not allowed

Error Type:
Warning

Possible Cause:
The CSV file being imported contains one or more Native Tags that have the same Logix Address.

Solution:
In the Native Tag Database CSV file being imported, eliminate Native Tags that contain a duplicate Logix Address.

Index