Omron NJ Ethernet Driver

Table of Contents

Omron NJ Ethernet Driver

Help version 1.022

CONTENTS

Overview

The Omron NJ Ethernet Driver provides a reliable way to connect Omron NJ Ethernet controllers to client applications, including HMI, SCADA, Historian, MES, ERP, and countless custom applications.

Device Setup

Supported Controllers

Omron NJ 301
Omron NJ 501

Communication Protocol

Ethernet/IP (CIP over Ethernet) using TCP/IP.

Maximum Number of Channels and Devices

The maximum number of channels supported is 256. The maximum number of devices supported is 1024.

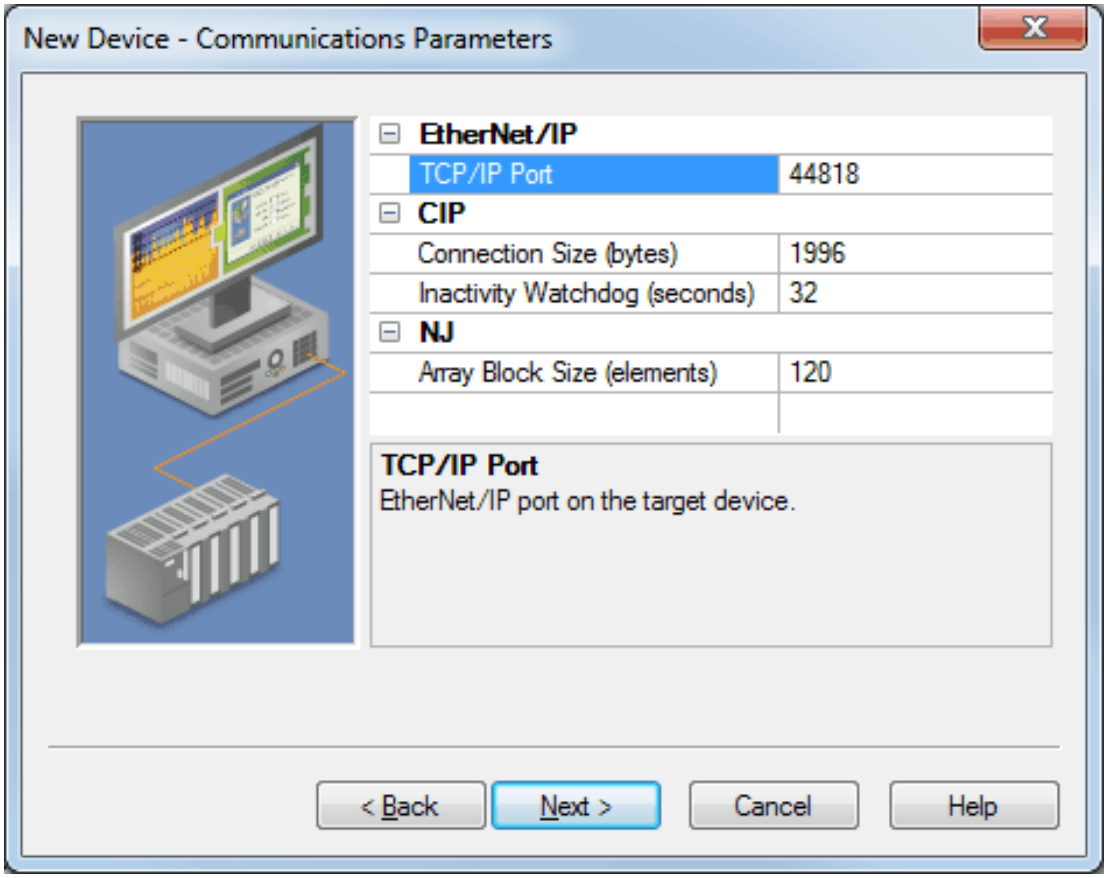
Device ID

The device ID specifies the path to the destination NJ CPU unit.

- The device ID for a local NJ CPU unit is specified as the IP or host name of the local CPU unit. It must be a valid dotted-quad IP Address or host name. For example, "192.168.1.100" or "NJ001".
- The device ID for a remote NJ CPU unit is specified as the IP or host name of the local CPU or Ethernet/IP unit plus a CIP Connection Path (also known as Routing Path) to the remote CPU unit. For example, "192.168.1.100\1\#10\2\10.10.110.2".

Note: For information on connection path syntax, refer to **Connection Path Specification**.

Communications Parameters

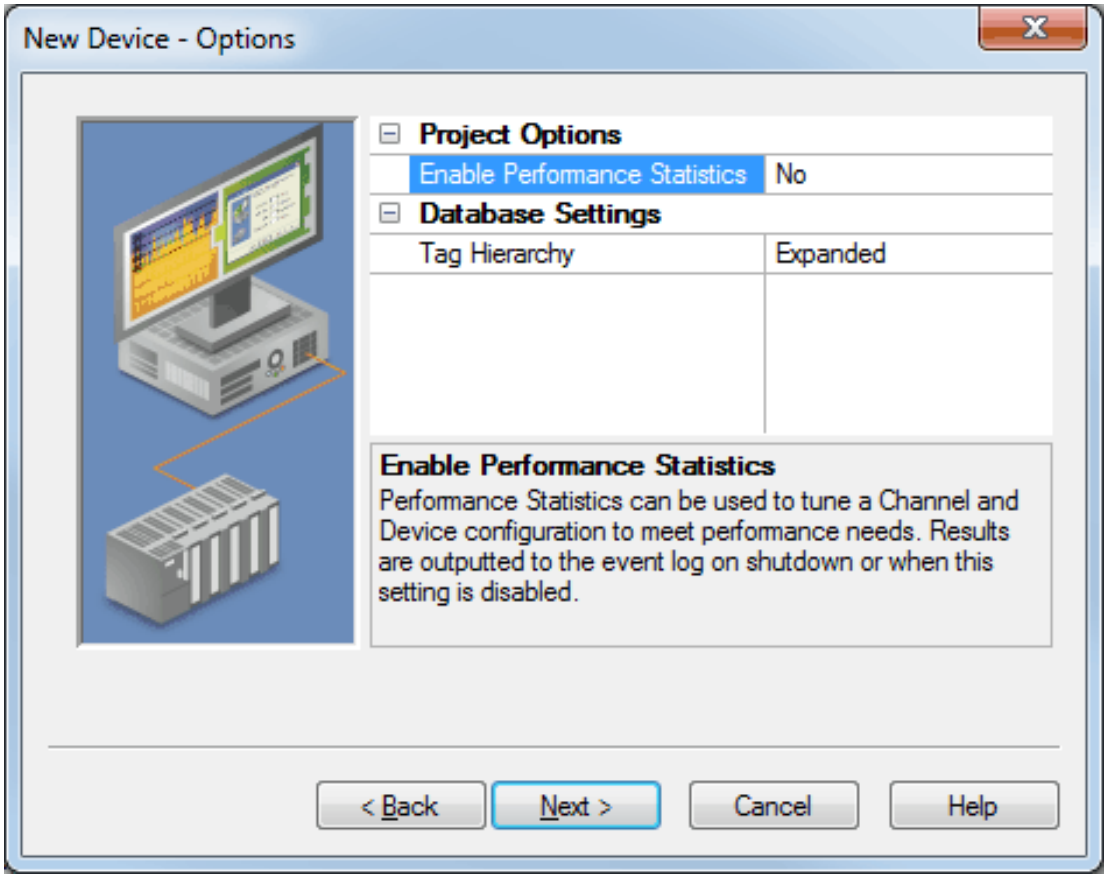


Descriptions of the parameters are as follows:

- **TCP/IP Port:** This parameter specifies the TCP/IP port number that the device is configured to use. The default setting is 44818.
- **Connection Size:** This parameter specifies the number of bytes available on the CIP connection for data requests and responses. The valid range is 500 to 1996 bytes. The default setting is 1996 bytes.

  Important: The Connection Size value may also be requested through the System Tag "_CIPConnectionSizeRequested."
- **Inactivity Watchdog:** This parameter specifies the amount of time a connection can remain idle (without Read/Write transactions) before being closed by the controller. In general, the larger the watchdog value, the more time it will take for connection resources to be released by the controller and vice versa. The default setting is 32 seconds.
- **Array Block Size:** This parameter specifies the maximum number of array elements to read in a single transaction. The value is adjustable and ranges from 30 to 3840 elements. The default setting is 120 elements.

Options



Descriptions of the parameters are as follows:

- **Enable Performance Statistics:** The Omron NJ Ethernet Driver has the ability to gather communication statistics to help determine the driver's performance. When checked, this option will be enabled. The driver will then track the number and types of Client/Server Tag updates. On restart of the server application, the results will be displayed in the server's Event Log. The default setting is No.

  Note: Once a project configuration is designed for optimal performance, it is recommended that users disable Performance Statistics. Furthermore, since the statistics are outputted to the Event Log on shutdown, the server will need to be re-launched to view the results.
- **Tag Hierarchy:** This parameter specifies the tag hierarchy. Options include Condensed and Expanded. The default setting is Expanded. Descriptions of the options are as follows:

  - **Condensed Mode:** In this mode, the server tags created by automatic tag generation follow a group/tag hierarchy consistent with the tag's address. Groups are created for every segment preceding the period.
  - **Expanded Mode:** In Expanded Mode, tag groups are created for every segment preceding the period (as in Condensed Mode), but groups are also created for array tags. This is the default setting.

    Note: For more information on the groups that are created, refer to **Tag Hierarchy**.

Note: To enable this functionality, check **Allow Automatically Generated Subgroups** in Device Properties.

Communications Routing and Timing

Routing provides a way to communicate with a remote NJ CPU via a local NJ Ethernet/IP unit. For more information on routing, refer to Chapter 8 of Omron's NJ-series CPU Unit Built-in Ethernet/IP Port User's Manual W506. Although the material focuses on NJ to NJ communications, the same concepts apply when the driver is communicating with a remote NJ CPU.

Routing Timing

When communications with a remote CPU are lost, the driver will utilize different request timeout parameters than configured in the device's timing settings when performing the following:

- **Identity requests:** Unconnected messages used to determine the remote CPU's model and Firmware version.
- **Forward Open requests:** Unconnected messages used to establish a high-level CIP connection with the remote CPU.

In these situations, the local device will return CIP Error 0x01 Ext. Err 0x204, which is defined as an "unconnected request timeout" and indicates that the remote CPU could not be reached. These requests will usually occur after a connection has been closed following a read or write request timeout. In this scenario, the device may enter an error state quickly following a read or write request timeout but fail tags slowly thereafter as it waits for the unconnected request timeout to occur. Although the driver will wait the entire "unconnected request timeout," it is possible for the local device to respond sooner with this error.

This custom timeout is based on the number of segments in the routing path. Only one attempt is made per request. For more information, refer to the table below.

Number of Segments	Request Timeout (s)	Example
1	15	10.10.110.2\1\0
2	20	10.10.110.2\1\#10\2\172.16.1.3
3	25	10.10.110.2\1\#10\2\172.16.1.3\1\0
4	30	10.10.110.2\1\#10\2\192.168.1.3\1\0\2\172.16.1.4
5	35	10.10.110.2\1\#10\2\192.168.1.3\1\0\2\172.16.1.4\1\0

Note: The driver will always utilize the device's timing settings when performing reads, writes, and Automatic Tag Generation operations regardless of whether routing to a remote CPU or directly to a local CPU. For more information, refer to "Device Properties - Timing" in the server help documentation.

Connection Path Specification

The CIP connection path (more commonly known as the routing path) is specified in the Device ID. Communication originates from the Omron NJ Ethernet Driver on the PC and is directed at the local NJ CPU or Ethernet/IP unit. Once at this local unit, the Device ID specifies a way out of the unit and onto the back plane. The routing path then directs the message to the desired remote NJ CPU unit.

The routing path itself is a series of Port/Link Address pairs, which are identical to the routing paths described in Chapter 8 of Omron's NJ-series CPU Unit Built-in Ethernet/IP Port User's Manual W506. In that document, "Network type number" is synonymous with Port and "Remote address" is synonymous with Link Address. Within the routing path, both Ports and Link Addresses are delimited by a backslash (no spaces necessary).

Note: A routing path is not necessary when the destination is the local NJ CPU unit. The Device ID only needs to contain the IP address of the local CPU unit.

Designator Type	Description	Formats	Range
Port (Network type number)	Specifies a way out of the interface unit in question. Back plane (BP) Port: 1 Ethernet/IP (EIP) Port: 2	Decimal Hex	0-65535 #0000-#FFFF
Link Address (Remote address)	Specifies a destination from the Port. Back plane (BP) Port: Unit Address of the destination unit Ethernet/IP (EIP) Port: IP Address of the remote CPU or Ethernet/IP (EIP) unit	Decimal Hex	Unit: 0-255 IP: Valid dotted-quad IP* #00-#FF

*Host names are not allowed.

Device ID Syntax containing Routing Path

Local CPU (0 Hops)
Local EIP IP \ BP Port \ CPU Unit Address

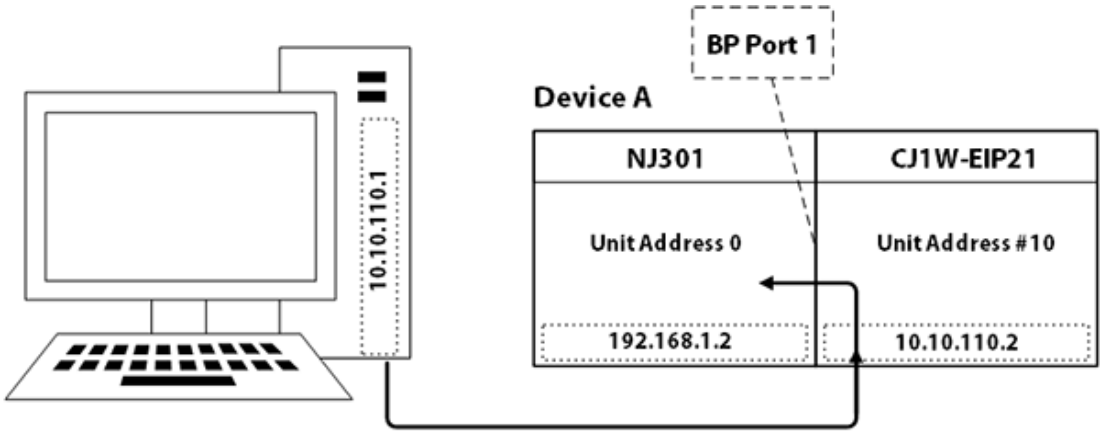
Remote CPU (1 Hop)
Local CPU IP \ BP Port \ EIP Unit Address \ EIP Port \ Remote CPU IP Address or Local CPU IP\ BP Port \ EIP Unit Address \ EIP Port \ Remote EIP IP Address\ BP Port\ Remote CPU Unit Address

Multi-Hop (*N* Hops)
Local CPU IP \ BP Port \ EIP Unit Address \ EIP Port \ Remote CPU IP Address \ BP Port… \ EIP Unit Address \ EIP Port \ Remote CPU IP Address or Local CPU IP \ BP Port \ EIP Unit Address \ EIP Port \ Remote CPU IP Address \ BP Port… \ EIP Unit Address \ EIP Port \ Remote EIP IP Address \ BP Port \ Remote CPU Unit Address

Routing Examples

The routing examples below include the entire Device ID.

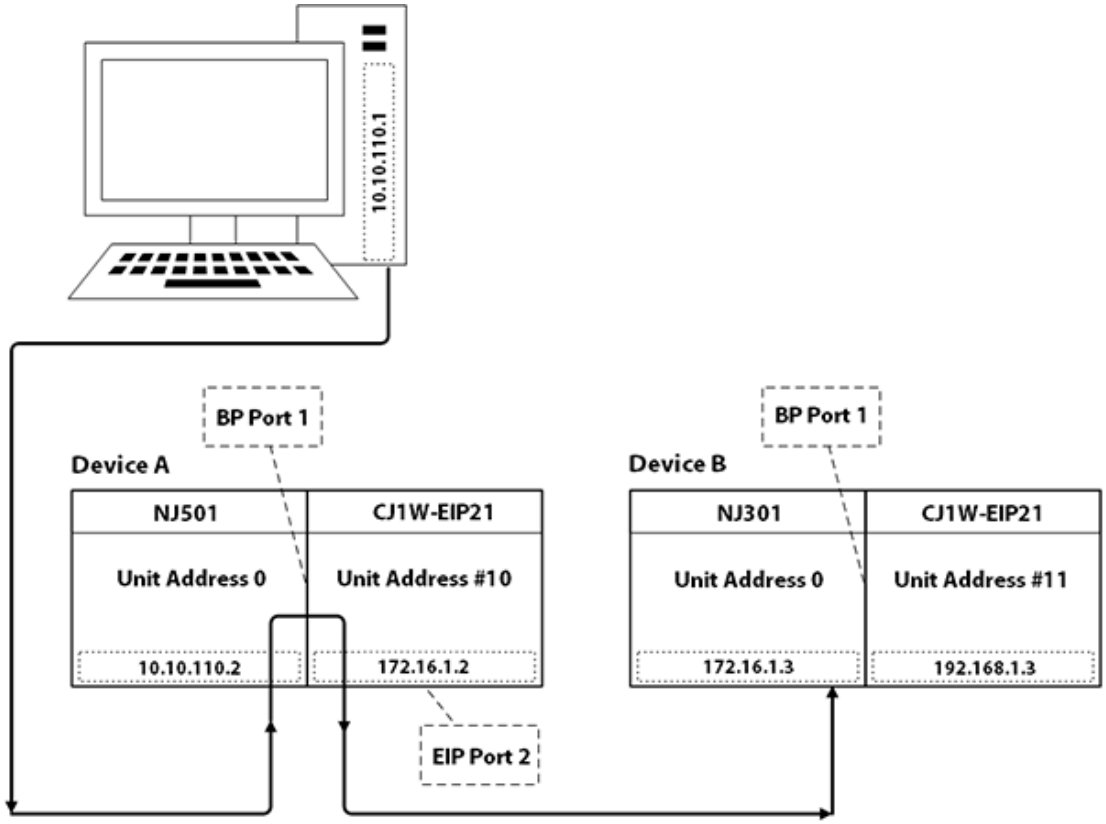
Local Omron NJ (A): 10.10.110.2\1\0



The breakdown of the *10.10.110.2\1\0* path is as follows:

- **10.10.110.2:** IP Address of Device A's EIP unit
- **\1:** Port # of Device A's EIP unit to access Back plane
- **\0:** Unit Address of Device A's CPU unit

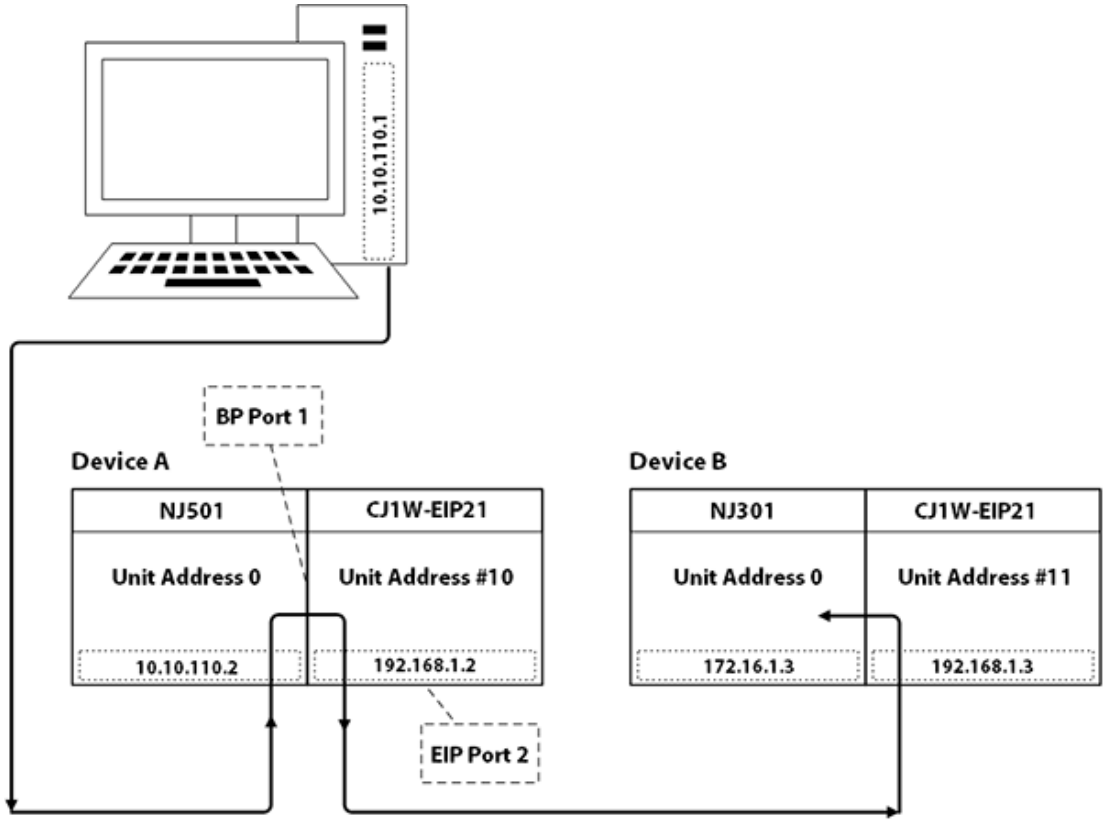
Remote Omron NJ (B) via Omron NJ (A): 10.10.110.2\1\#10\2\172.16.1.3



The breakdown of the *10.10.110.2\1\#10\2\172.16.1.3* path is as follows:

- **10.10.110.2:** IP Address of Device A's CPU unit
- **\1:** Port # of Device A's CPU unit to access Back plane
- **\#10:** Unit Address of Device A's EIP unit (10 hex, 16 dec)
- **\2:** Port # of Device A's EIP unit to access Ethernet/IP
- **\172.16.1.3:** IP Address of Device B's CPU unit

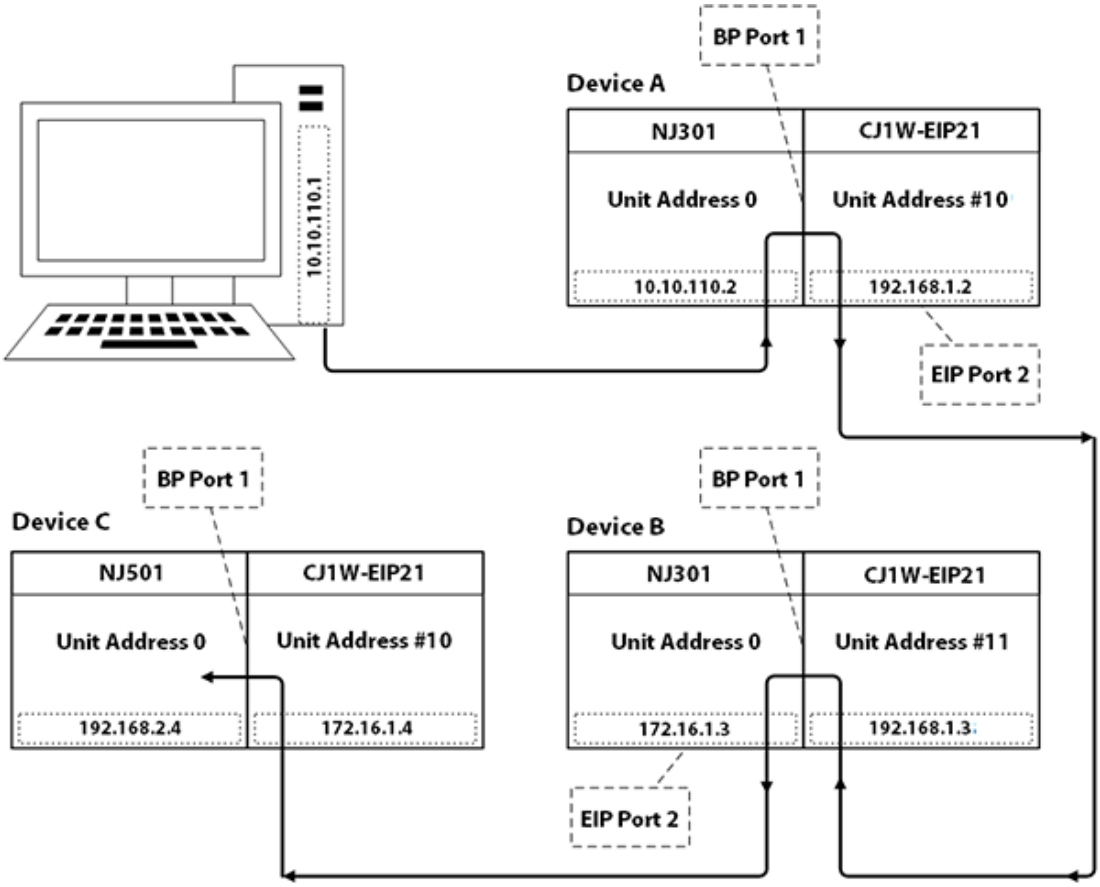
Remote Omron NJ (B) via Omron NJ (A): 10.10.110.2\1\#10\2\192.168.1.3\1\0



The breakdown of the *10.10.110.2\1\#10\2\192.168.1.3\1\0* path is as follows:

- **10.10.110.2:** IP Address of Device A's CPU unit
- **\1:** Port # of Device A's CPU unit to access Back plane
- **\#10:** Unit Address of Device A's EIP unit (10 hex, 16 dec)
- **\2:** Port # of Device A C1J-EIP21 unit to access Ethernet/IP
- **192.168.1.3:** IP Address of Device B's EIP unit
- **\1:** Port # of Device B's EIP unit to access Back plane
- **\0:** Unit Address of Device B's CPU unit

Remote Omron NJ (C) via Omron NJ (A): 10.10.110.2\1\#10\2\192.168.1.3\1\0\2\172.16.1.4\1\0



The breakdown of the *10.10.110.2\1\#10\2\192.168.1.3\1\0\2\172.16.1.4\1\0* path is as follows:

- **10.10.110.2:** IP Address of Device A's CPU unit
- **\1:** Port # of Device A's CPU unit to access Back plane
- **\#10:** Unit Address of Device A's EIP unit (10 hex, 16 dec)
- **\2:** Port # of Device A's EIP unit to access Ethernet/IP
- **\192.168.1.3:** IP Address of Device B's EIP unit
- **\1:** Port # of Device B's EIP unit to access Back plane
- **\0:** Unit Address of Device B's CPU unit
- **\2:** Port # of Device B's CPU unit to access Ethernet/IP
- **\172.16.1.4:** IP Address of Device C's EIP unit
- **\1:** Port # of Device C's EIP unit to access Back plane
- **\0:** Unit Address of Device C's CPU unit

Note: For more information, refer to **Connection Path Specification**. For more information on building a connection/routing path, refer to Chapter 8 of Omron's NJ-series CPU Unit Built-in Ethernet/IP Port User's Manual W506.

Optimizing Communications

As with any programmable controller, there are a variety of ways to enhance the performance and system communications.

Connection Size

Increasing the Connection Size allows more Read/Write requests per data packet, which provides greater throughput. Although it also increases the CPU load and response turnaround time, it significantly improves performance. For more information, refer to **Communications Parameters**.

Multi-Request Packets

The Omron NJ Ethernet Driver has been designed to optimize reads and writes by including multiple requests in a single transaction. This provides drastic improvement in performance over single tag transactions. The only limitation is the number of data bytes that can fit in a single transaction.

Important: Because read and write requests specify variables' addresses in ASCII format, users should keep the size of the variables' names to a minimum. The smaller the variable name, the more tags that will fit in a single transaction, and the fewer transactions needed to process all tags.

Blocking Array Elements

To optimize the reading of basic array elements, read a block of the array in a single request instead of individually. The more elements read in a block, the greater the performance. Since transaction overhead and processing consumes the most time, do as few transactions as possible while scanning as many desired tags as possible. This is the essence of array element blocking.

Block sizes are specified as an element count. A block size of 120 elements means that a maximum of 120 array elements will be read in one request. The maximum block size of 3840 elements means a maximum of 3840 array elements will be read in one request.

As discussed in **Communication Parameters**, the block size is adjustable and should be chosen based on the project at hand. For example, if array elements 0 to 26 and element 3839 are tags to be read, then using a block size of 3840 is not only overkill, but detrimental to the driver's performance. This is because all elements between 0 and 3839 will be read on each request, even though only 28 of those elements are of importance. In this case, a block size of 30 is more appropriate. Elements 0 to 26 would be serviced in one request and element 3839 would be serviced on the next.

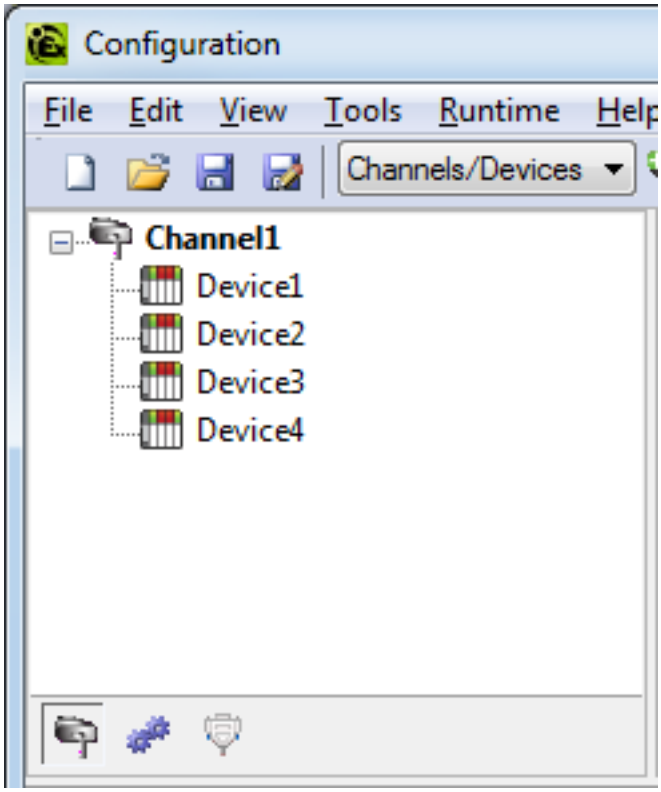
Strings

It is recommended that string variables be defined with the smallest string length necessary to serve their purpose. In Sysmac Studio, string variables are defined with a length of 256 by default. Reading these string variables with large string lengths will incur extra device communications and may affect performance.

Optimizing the Application

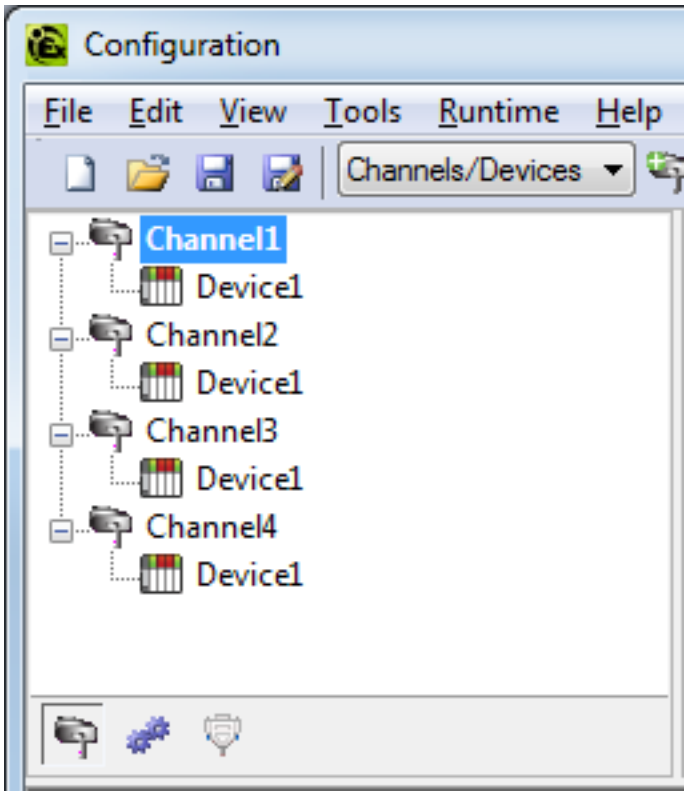
The Omron NJ Ethernet Driver has been designed to provide the best performance with the least amount of impact on the system's overall performance. While the Omron NJ Ethernet Driverr is fast, there are a couple of guidelines that can be used in order to optimize the application and gain maximum performance.

The server refers to communications protocols like Omron NJ Ethernet 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 must then be defined under that channel. Each of these devices represents a single Omron NJ CPU 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 Omron NJ Ethernet Driver or the network. An example of how the application may appear when configured using a single channel is shown below.



Each device appears under a single channel, called "Channel1". In this configuration, the driver must move from one device to the next as quickly as possible in order to gather information at an effective rate. As more devices are added or more information is requested from a single device, the overall update rate begins to suffer.

If the Omron NJ Ethernet Driver could only define one channel, then all devices needed for the project would have to be created beneath it; however, the driver can define up to 256 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 256 or fewer devices, it can be optimized exactly how it is shown here.

The performance will improve even if the application has more than 256 devices. While 256 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 less devices to process on a single channel.

Performance Statistics and Tuning

The Performance Statistics feature provides benchmarks and statistics about the Omron NJ Ethernet application's performance. It can affect the server's performance because it is an additional layer of processing. As such, it is disabled by default. To enable the Performance Statistics feature, open **Device Properties** and select the **Options** tab. In **Enable Performance Statistics**, select **Yes**.

Types of Performance Statistics

Performance Statistics provide meaningful numerical results across three scopes: device, channel, and driver. Descriptions of the types are as follows:

- **Device:** These statistics provide the data access performance on a particular device.
- **Channel:** These statistics provide the average data access performance for all the devices under a given channel with Performance Statistics enabled.
- **Driver:** These statistics provide the average data access performance for all devices using the Omron NJ Ethernet Driver with Performance Statistics enabled.

Choosing a Statistic Type

The type of statistics needed depends on the application. In general, driver statistics provide a true measure of the application's performance, whereas channel and device statistics are most relevant while tuning the application. For example, will moving 10 certain tags from Device A to Device B increase the performance of Device A? Will moving Device A from Channel 1 to Channel 2 increase the performance of Channel 1? These questions are good examples of situations when device and channel statistics should be used.

Locating Statistics

Server statistics are outputted to the server's Event Log upon shutdown. To view the results, shut down the server and then restart it.

Differences between Server Statistics and Performance Statistics

Performance Statistics provide the makeup of the types of reads performed (such as device reads vs. cache reads) whereas server statistics provide a general read count value.

Tuning the Application for Increased Performance

To increase device and channel statistic results, keep variable names to a minimum length and use Variable Arrays as often as possible. For more information, refer to **Optimizing Communications**.

For information on increasing driver statistic results, refer to the instructions below. For more information, refer to **Optimizing the Application**.

1. Devices should be spread across channels. More than one device should not be put on a channel unless necessary.
2. Load should be spread evenly across devices. A single device should not be overloaded unless necessary.
3. The same Variable Tag should not be referenced across different devices.

Data Type Descriptions

Data Type	Description
Boolean	Single bit
Byte	Unsigned 8 bit value
Char	Signed 8 bit value
Word	Unsigned 16 bit value bit 0 is the low bit bit 15 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
Long	Signed 32 bit value
DWord	Unsigned 32 bit value
Float	32 bit IEEE floating point
Double	64 bit IEEE floating point
Date	64 bit date and time value
String	Null terminated Unicode string
Default	*

*If the data type is specified as "Default" when creating a Static Tag, the driver will query the controller for the tag's data type and then set the canonical data type for items referencing that Static Tag to the query result. If a data type is not specified when creating a Dynamic Tag, the driver will query the controller for the tag's data type and then set the canonical data type for that Dynamic Tag to the query result.

Address Descriptions

The Omron NJ Ethernet Driver uses a tag or symbol-based addressing structure referred to as variables. These tags differ from conventional PLC data items in that the tag name itself is the address, not a file or register number. Users can access the controller's basic data types. Although some of the system-defined types are structures, they are ultimately based on these basic data types. Thus, all basic members of a structure are accessible.

Omron Data Type	Description	Data Type	Range
BOOL	Single bit value	Boolean	0, 1
SINT	Signed 8 bit value	Char	-128 to 127
USINT	Unsigned 8 bit value	Byte	0 to 255
BYTE	Bit string (8 bits)	Byte	0 to 255
INT	Signed 16 bit value	Short	-32768 to 32767
UINT	Unsigned 16 bit value	Word	0 to 65535
WORD	Bit string (16 bits)	Word	0 to 65535
DINT	Signed 32 bit value	Long	-2147483648 to 2147483647
UDINT	Unsigned 32 bit value	DWord	0 to 4294967295
DWORD	Bit string (32 bits)	DWord	0 to 4294967295
LINT	Signed 64 bit value	Double	-9223372036854775808 to 9223372036854775807
ULINT	Unsigned 64 bit value	Double	0 to 18446744073709551615
REAL	32 bit IEEE floating point	Float	-3.402823e+38 to -1.175495e-38 0 1.175495e-38 to 3.402823e+38
LREAL	64 bit IEEE floating point	Double	-1.79769313486231e+308 to -2.22507385850721e-308 0 2.22507385850721e-308 to 1.79769313486231e+308
DATE AND TIME	Unsigned 64 bit value	Date	The date/time variable format is: YYYY-MM-DDTHH:MM:SS.MS. The supported range is 1970-01-01T00:00:00.000 to 2106-02-06T23:59:59.999.
STRING	Character string	String	String lengths range from 1 to 1985 characters. This equates to variables defined in Sysmac Studio as STRING[2] and STRING[1986] respectively. The extra character accounts for the null terminator.
Enumeration	Signed 32 bit value	Long	-2147483648 to 2147483647*

*The valid values for an enumeration are actually a subset of the values in the specified range. The subset of values is determined by the configuration of the enumeration in the Omron NJ device.

Client/Server Tag Address Rules

Variable names correspond to Client/Server Tag addresses. Both variable names and Client/Server Tag addresses follow the IEC 61131-3 identifier rules. Descriptions of the rules are as follows:

- Can only contain alphanumeric characters and underscores
- Can have as many as 127 characters per segment
- Characters are not case sensitive
- White spaces are ignored

Client/Server Tag Name Rules

Tag name assignment in the server differs from address assignment in that names cannot begin with an underscore. For syntax and examples, refer to **Address Formats**.

Important: If a tag address is large enough that it exceeds the protocol limit of 511 bytes, it will fail validation with an "Address out of range" error. If this occurs, reduce the number of characters in the tag address until it passes validation.

Data Type Coercion

The Omron NJ Ethernet Driver can coerce some Omron data types in the controller to more than one server data type. For example, the tag for a SINT variable in the controller can be created with a Byte server data type. For a list of the supported data type coercions for all Omron data types that are supported by the driver, refer to the table below.

Omron Data Type	Data Type
BOOL	Boolean
SINT, USINT, OR BYTE	Char or Byte
INT, UINT, OR WORD	Short or Word
DINT, UDINT, DWORD, OR ENUM	Long or DWord
LINT OR ULINT	Double
REAL	Float
LREAL	Double
DATE AND TIME	Date
STRING	String

Address Formats

A Variable Tag may be addressed statically in the server or dynamically from a client in several ways. The tag's format will depend on its type and intended usage. Descriptions of the variable types are as follows:

- **Array Element:** A variable may be defined in the controller using the following syntax: *ARRAY [x1..x2, y1..y2, z1..z2] OF TYPE*, where *TYPE* is one of the Omron data types listed in **Address Descriptions**. To access individual elements, specify the *x*, *y*, and *z* offsets. The driver blocks read requests on the last dimension. For example, with a variable like "MyArray[1,0]" and "MyArray[1,4]," the driver will perform a single request for five elements starting at "MyArray[1,0]." For more information, refer to **Communication Parameters** and **Optimizing Communications**.
- **Array:** A variable may be defined in the controller using the following syntax: *ARRAY [x1..x2, y1..y2, z1..z2] OF TYPE*, where *TYPE* is one of the Omron data types listed in **Address Descriptions**. To access multiple elements in a single client item, use the array type syntax. Like Array Elements, the driver will perform a single request to read and write multiple array elements. The difference with arrays is that all items in the array will be provided to the client in an atomic operation. String Arrays are not supported.

  Note: Not all clients support array types. For support information, refer to the client application.
- **Basic:** A variable defined with a basic type and no array syntax.
- **String:** A variable defined with the string basic type.

Note: All Symbolic Variable Tag names in Sysmac Studio can be copied and pasted into the server's tag address field and be valid.

Array Element

At least one dimension (but no more than three) must be specified.

Syntax	Example
<Variable Tag name> [dim1]	tag_1 [5]
<Variable Tag name> [dim 1, dim2]	tag_1 [2, 3]
<Variable Tag name> [dim 1, dim2, dim 3]	tag_1 [2, 58, 547]

Examples
MyBooleanArray[31]
MyBooleanArray3D[2,2,7]
MySintArray[1]
MyLrealArray[65535]
MySintArray2D[1,2]
MyLrealArray2D[2,500]
MySintArray3D[2,3,9]
MyLrealArray3D[2,10,10]

Array

With this format, multiple elements of a Variable Array are read and written in a single transaction. The client must support array types (such as "VT_ARRAY"). Client data is organized in a row by column format to facilitate one dimension (1 row, *y* columns) or two dimensions (*x* rows, *y* columns). This format is supported for one-

dimensional, two-dimensional, and three-dimensional Variable Arrays only. Like Array Elements, at least one dimension (but no more than three) must be specified.

Note: All Omron data types that are supported by this driver support the array format except string and date and time.

Important: Spanning an array across multiple Variable Array dimensions is not supported. If an array is created on a two-dimensional or three-dimensional Variable Array, the size of that array (which is rows by columns) must not exceed the bounds of the last dimension. For example, given a Variable Array "MySintArray3D" defined as ARRAY[0..2,0..3,0..9] OF SINT, the Array Tag MySintArray3D[0,0,0]{10} is valid because it references elements [0,0,0..9] that all lie within the last dimension. MySintArray3D[0,0,0]{11} is invalid, however, because it is attempting to reference elements [0,0,0..9] and [0,1,0] that exceed the bounds of the last dimension by one element.

Syntax	Example
<Variable Tag name> [dim 1 offset] {# of columns}	tag_1 [5]{8}
<Variable Tag name> [dim 1 offset, dim 2 offset] {# of columns}	tag_2 [0, 5]{8}
<Variable Tag name> [dim 1 offset, dim 2 offset, dim 3 offset] {# of columns}	tag_3 [1,0, 5]{8}
<Variable Tag name> [dim 1 offset] {# of rows}{# of columns}	tag_4 [5]{2}{4}
<Variable Tag name> [dim 1 offset, dim 2 offset] {# of rows}{# of columns}	tag_1 [0,5]{2}{4}
<Variable Tag name> [dim 1 offset, dim 2 offset, dim 3 offset] {# of rows}{# of columns}	tag_1 [1,0,5]{2}{4}

Note: The number of elements to read and/or write equals the number of rows multiplied by the number of columns. If no rows are specified, the number of rows will default to 1. At least one element of the array must be addressed. Rows x Columns must be between 1 and 65535.

Examples
MyBooleanArray[0]{32}
MyBooleanArray3D[2,2,7]{1}
MySintArray[1]{5}
MyLrealArray[65535]{1}
MySintArray2D[1,2]{10}
MyLrealArray2D[2,500]{14}{20}
MySintArray3D[2,3,9]{10}
MyLrealArray3D[2,10,10]{14}{20}

Basic

Syntax	Example
<Variable Tag name>	tag_1

Examples
MyBool
MyByte
MyInt
MyWord
MyReal

String

The number of characters to read and/or write equals the string length, which must be at least 2. Although Sysmac Studio allows a variable to be defined as STRING [1], reads and writes will not be possible because one character is reserved for a null terminator. As such, it is recommended that string variables be defined with a length of 2 to 1986. To account for this null terminator, the valid range for string lengths in the driver is 1 to 1985. Strings support any character encoded in UTF-8. One UTF-8 character can equal 1 to 4 bytes.

Note: A 256 byte string containing characters that require multiple bytes when encoded in UTF-8 represents fewer than 256 characters.

Syntax	Example
<Variable Tag name> / <string length>	tag_1 / 255

Examples
MyString256/255
MyString1986/1985
MyString1986/100

MyStruct[23].Banners[4].Output/10
MyStringArray3D[2,10,10]/255

Tag Scope

Note: Local variables can only be read and written in a POU (program, function, or function block) in which it is defined.

Global Tags

Global Tags are Variable 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 both its Variable Data Type and the address format being used.

Structure Tag Addressing

Structure Tags are tags with one or more member tags, which can be basic or structured in nature.

<structure name> . <basic-type tag>

This implies that a substructure would be addressed as:

<structure name> . <substructure name> .<basic-type tag>

Arrays of structures would be addressed as follows:

<structure array name> [dim1, dim2, dim3] . <basic-type tag>

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

<structure name> . <substructure array name> [dim1, dim2, dim3] . <basic-type tag>

Note: The examples given above are only a few of the many addressing possibilities involving structures. They are displayed in order to provide an introduction to structure addressing. For more information, refer to Omron NJ documentation.

Predefined Term Tags

The tags displayed in the table below can be used to obtain general processor information from a PLC.

Tag Name	Data Type	Description
#DEVICETYPE	Word	An integer value that corresponds to the "ProdType" attribute specified in the PLC's EDS file.
#REVISION	String	Firmware revision displayed as *<major>.<minor>*.
#PRODUCTNAME	String	The processor name that corresponds to the "ProdName" attribute specified in the PLC's EDS file.
#PRODUCTCODE	Word	An integer value that corresponds to the "ProdCode" attribute specified in the PLC's EDS file.
#VENDORID	Word	An integer value that corresponds to the "VendCode" attribute specified in the PLC's EDS file.

Automatic Tag Database Generation

The Omron NJ Ethernet Driver can be configured to automatically generate a list of tags within the server that correspond to the Global Variables used in the Omron SYSMAC NJ Series controller program and that are published to the network as inputs, outputs, or publish-only variables.

To generate tags from the device:

1. In the Configuration, select the device for which tags will be generated.
2. Right-click and select **Properties...** to open the Device Properties dialog box.
3. Select the Database Creation tab.
4. Click the **Auto Create** button to initiate tag database creation.
5. Click the **Close** button to exit the dialog box.
6. Check the Event Log for messages confirming successful generation.

For more information about custom settings, see the server help file.

Notes:

1. It is recommended that all communications to the Omron NJ device cease during tag database creation process.
2. Variable tags generated for enumerations are data type Long.

See Also: **Address Formats** and **Address Descriptions**.

Tag Hierarchy

The server tags created by automatic tag generation can follow one of two hierarchies: Expanded or Condensed. To enable this functionality, ensure that "Allow Automatically Generated Subgroups" is enabled in Device Properties. The default setting is Expanded Mode.

Expanded Mode

In Expanded Mode, tag groups are created for every segment preceding the period (as in Condensed Mode), but are also created in logical groupings. Groups created include the following:

- Structures and substructures
- Unions
- Arrays

Basic Global tags (or non-structure, non-union, and non-array tags) are placed at the device level. Each structure, union, and array tag is provided in its own subgroup of the parent group.

The name of the structure, union, or array subgroup also provides a description of the structure, union, or array. For example, an array tag1[1,6] defined in the controller would have a subgroup name of "tag1[*x*,*y*]" where *x* signifies dimension 1 exists and *y* signifies dimension 2 exists. The tags within an array subgroup are all the elements of that array. The tags within a structure subgroup are the structure members themselves. If a structure contains an array, then an array subgroup of the structure group will be created as well. The tags within a union subgroup are the union members themselves. If a union contains an array, then an array subgroup of the union group will be created as well.

Array Tag Groups

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

- [*x*,*y*,*z*] is a 3 dimensional array
- [*x*,*y*] is a 2 dimensional array
- [*x*] is a 1 dimensional array

Array Tags will have the notation: <*tag element*>[*XXXXX*,*YYYYY*,*ZZZZZ*]. For example, element tag1[12,2,987] would have the tag name "tag1[12,2,987]".

Condensed Mode

In Condensed Mode, the server tags created by automatic tag generation follow a group/tag hierarchy consistent with the tag's address. Groups are created for every segment preceding the period. Groups created include the following:

- Structures and substructures
- Unions

Note: Groups are not created for arrays.

Error Descriptions

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

Address Validation Messages

The following 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.

Error Type:
Warning

Possible Cause:

1. A tag address that has been specified statically references a location that is beyond the range of supported locations for the device.
2. The tag address size exceeds the protocol limits.

Solution:

1. Verify the address is correct; if it is not, re-enter it in the client application.
2. Reduce the number of characters in the tag name. If the error persists, reduce the number of characters in the structure or union (including any nested structures or unions) under which the tag is located.

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 for the address type or block size of the driver.

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

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 doesn't support arrays.

Solution:

Re-enter the address in the client application to remove the array reference or correct the address type.

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

Error Type:
Warning

Possible Cause:
A tag address 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 invalid characters.

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

Device address <address> is not supported by model <model name>.

Error Type:
Warning

Possible Cause:
A tag address that has been specified statically references a location that is valid for the communications protocol but not supported by the target device.

Solution:
Verify the address is correct; if it is not, re-enter it in the client application. Also verify that the selected model name for the device is correct.

Device address <address> is read only.

Error Type:
Warning

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

Solution:
Change the access mode in the client application.

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 Messages

The following is a list of sub type error topics. Click on a link for more information about that specific error message.

Database Error: Address validation failed for tag <tag name> with address <tag address>. Tag will not be added to the database.

Error Type:
Warning

Possible Cause:
The tag address size exceeds the protocol limits.

Solution:
Reduce the number of characters in the tag name. If the error persists, reduce the number of characters in the structure or union (including any nested structures or unions) under which the tag is located.

Database Error: Data type <hex value> for member <member name> of complex type <complex type name> is not supported. A tag for this member will not be added to the database.

Error Type:
Warning

Possible Cause:
The device's configuration of the complex type's member uses an unsupported data type.

Solution:
Modify the device's configuration of the complex type's member to use a supported data type.

See Also:
Address Descriptions

Database Error: Encapsulation error occurred during Fwd. Open request. [Encap. Error=<error>].

Error Type:
Error

Possible Cause:
The device returned an error within the encapsulation portion of the Ethernet/IP packet during an automatic tag generation request.

Solution:
The driver attempts to recover from this error. If the problem persists, contact Technical Support.

Note:

This excludes error 0x02, which is device-related and not driver-related.

See Also:
Encapsulation Error Codes

Database Error: Encapsulation error occurred during Register Session request. [Encap. Error=<error>].

Error Type:
Error

Possible Cause:
The device returned an error within the encapsulation portion of the Ethernet/IP packet during an automatic tag generation request.

Solution:
The driver will attempt to recover from this error. If the problem persists, contact Technical Support.

Note:
This excludes error 0x02, which is device-related and not driver-related.

See Also:
Encapsulation Error Codes

Database Error: Encapsulation error occurred while uploading project information. [Encap. Error=<error>].

Error Type:
Error

Possible Cause:
The device returned an error within the encapsulation portion of the Ethernet/IP packet during an automatic tag generation request.

Solution:
The driver will attempt to recover from this error. If the problem persists, contact Technical Support.

Note:
This excludes error 0x02, which is device-related and not driver-related.

See Also:
Encapsulation Error Codes

Database Error: Error occurred during Fwd. Open request [CIP Error=<code>, Ext. Error=<code>].

Error Type:
Error

Possible Cause:
The device returned an error within the CIP portion of the Ethernet/IP packet during an automatic tag generation request.

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

See Also:
CIP Error Codes

Database Error: Error occurred while uploading project information. [CIP Error=<code>, Ext. Error=<code>].

Error Type:

Error

Possible Cause:
The device returned an error within the CIP portion of the Ethernet/IP packet during an automatic tag generation request.

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

See Also:
CIP Error Codes

Database Error: Framing error occurred during Fwd. Open request.

Error Type:
Error

Possible Cause:

1. The packets are misaligned due to the connection and/or disconnection between the PC and device.
2. There is bad cabling connecting the device that is causing noise.

Solution:

1. Place the device on less noisy network.
2. Increase the request timeout and/or request attempts.

Database Error: Framing error occurred during Register Session request.

Error Type:
Error

Possible Cause:

1. The packets are misaligned due to the connection and/or disconnection between the PC and device.
2. There is bad cabling connecting the device that is causing noise.

Solution:

1. Place the device on less noisy network.
2. Increase the request timeout and/or request attempts.

Database Error: Framing error occurred while uploading project information.

Error Type:
Error

Possible Cause:

1. The packets are misaligned due to the connection and/or disconnection between the PC and device.
2. There is bad cabling connecting the device that is causing noise.

Solution:

1. Place the device on less noisy network.
2. Increase the Request Timeout and/or Request Attempts.

Database Error: No more connections available for Fwd. Open request.

Error Type:
Error

Possible Cause:
Omron devices support a finite number of connections. The connection limit has been exceeded.

Solution:
Reduce the number of connections from the server(s) to the device and try again.

Database Error: Unable to resolve CIP data type <hex value> for tag <tag name>. Tag will not be added to the database.

Error Type:
Warning

Possible Cause:

1. A communications error occurred during automatic tag database generation that was caused by Ethernet encapsulation, the device, or framing.
2. The CIP data type is not supported by the driver.

Solution:

1. Correct the communications error and then retry automatic tag database generation.
2. Change the variable data type to one that is supported.

Unable to generate a tag database for device <device name>. Reason: Low memory resources.

Error Type:
Warning

Possible Cause:
Memory required for database generation could not be allocated. The process is aborted.

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

Communication Messages

The following is a list of sub type error topics. Click on a link for more information about that specific error message.

Unable to bind to adapter: <adapter>. Connect failed.
Winsock initialization failed (OS Error = n).
Winsock V1.1 or higher must be installed to use the Omron NJ Ethernet Driver.

Unable to bind to adapter: <adapter>. Connect failed.

Error Type:
Error

Possible Cause:
The driver was unable to bind to the specified network adapter, which is necessary for communications with the device.

Reasons:

1. The adapter is disabled or no longer exists.
2. A network system failure occurred (such as Winsock or network adapter failure).
3. There are no more available ports.

Solution:

1. For network adapters available on the system, check the Network Adapter list in the communications server application (located in Channel Properties). If the specified adapter is not in this list, steps should be taken to make it available to the system. This includes verifying that the network connection is enabled and connected in the PC's Network Connections.
2. Determine how many channels are using the same adapter in the communications server application. Then, reduce this number so that only one channel is referencing the adapter. If the error still occurs, check to see if other applications are using that adapter and then shut down those applications.

Winsock initialization failed (OS Error = n).

Error Type:
Error

OS Error:	Indication	Possible Solution
10091	The underlying network subsystem is not ready for network communication.	Wait a few seconds and restart the driver.
10067	The limit on the number of tasks supported by the Windows Sockets implementation has been reached.	Close one or more applications that may be using Winsock and restart the driver.

Winsock V1.1 or higher must be installed to use the Omron NJ Ethernet Driver.

Error Type:
Error

Possible Cause:
The version number of the Winsock DLL found on the system is less than 1.1.

Solution:
Upgrade Winsock to version 1.1 or higher.

Device-Specific Messages

The following is a list of device specific error topics. Click on a link for more information about that specific error message.

Device <device name> is not responding.
Encapsulation error occurred during a request to device <device name>. [Encap. Error=<code>].
Error occurred during a request to device <device name>. [CIP Error=<code>, Ext. Error=<code>].
Frame received from device <device name> contains errors.
Unable to retrieve the identity for device <device>. [CIP Error=<error>, Ext. Error=<error>].
Unable to retrieve the identity for device <device>. [Encap. Error=<error>].
Unable to retrieve the identity for device <device>. Frame received contains errors.

Device <device name> is not responding.

Error Type:
Warning

Possible Cause:

1. The Ethernet connection between the device and the Host PC is broken.
2. The communications parameters for the Ethernet connection are incorrect.
3. The named device may have been assigned an incorrect IP address.
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 device.
2. Verify that the correct port is specified for the named device.
3. Verify that the IP address given to the named device matches that of the actual device.
4. Increase the Request Timeout setting so that the entire response can be handled.

Encapsulation error occurred during a request to device <device name>. [Encap. Error=<code>].

Error Type:
Warning

Possible Cause:
The device returned an error within the encapsulation portion of the Ethernet/IP packet during a request. All reads and writes within the request failed.

Solution:
The driver attempts to recover from such an error. If the problem persists, contact Technical Support. This excludes error 0x02, which is device-related, not driver-related.

See Also:
Encapsulation Error Codes

Error occurred during a request to device <device name>. [CIP Error=<code>, Ext. Error=<code>].

Error Type:
Warning

Possible Cause:
The device returned an error within the CIP portion of the Ethernet/IP packet during a request. All reads and writes within the request failed.

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

See Also:
CIP Error Codes

Frame received from device <device name> contains errors.

Error Type:
Warning

Possible Cause:

1. The packets are misaligned due to connection and/or disconnection between the PC and device.
2. There is bad cabling connecting the device that is causing noise.

Solution:

1. Place the device on less noisy network.
2. Increase the request timeout and/or attempts.

Unable to retrieve the identity for device <device>. [CIP Error=<error>, Ext. Error=<error>].

Error Type:
Warning

Possible Cause:
The Identity was unable to be retrieved because the device returned an error within the CIP portion of the Ethernet/IP packet during a request.

Solution:
The solution depends on the error code that is returned. Consult the **CIP Error Codes**.

Unable to retrieve the identity for device <device>. [Encap. Error=<error>].

Error Type:
Warning

Possible Cause:
The Identity was unable to be retrieved because the device returned an error within the encapsulation portion of the Ethernet/IP packet during a request.

Solution:
The driver attempts to recover from such an error. If the problem persists, contact Technical Support. This excludes error 0x02, which is device-related, not driver-related.

See Also:
Encapsulation Error Codes

Unable to retrieve the identity for device <device>. Frame received contains errors.

Error Type:
Warning

Possible Cause:

1. The packets are misaligned due to connection and/or disconnection between the PC and device.
2. There is bad cabling connecting the devices that is causing noise.
3. The wrong frame size was received.
4. There is a TNS mismatch.
5. An invalid response command was returned from the device.
6. The device is not Ethernet/IP enabled.

Solution:

1. The driver will recover from this error without intervention. If this error occurs frequently, there may be an issue with the cabling, the network, or the device itself.
2. Verify that the device being communicated with is a Omron Ethernet/IP-enabled device.

Omron NJ Ethernet Messages

The following sections pertain to messaging from the Omron NJ Ethernet Driver source.

Read Errors (Non-Blocking)

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

Device <device name> returned more data than expected while reading tag <tag address>. Verify the address includes an element offset and all dimensions in that offset.

Error Type:
Warning

Possible Cause:
The tag address references an Array Variable, but either not all dimensions were specified in the element offset or no element offset was specified at all. For example, the Array Variable "MyArray" is defined as ARRAY [0..2,0..9] OF INT. Creating a tag address like MyArray[0]{2}@Word Array may cause this error to occur because the second dimension is not specified. In this example, a read of the tag would result in 10 INTs returned versus the 2 INTs that are expected. Likewise, creating a tag address like MyArray@Word may cause this error to occur because neither the first nor second dimension was specified. In this example, a read of the tag would result in 30 INTs returned versus the 1 INT that is expected.

Solution:
Add the fully-qualified element offset to the tag address. In the examples above, the correct tag address could be MyArray[0,0]{2} and MyArray[0,0] respectively.

Note:
This message is only a warning. Reads will succeed, but will be less efficient because of the extra overhead in the read response. As such, it is recommended that users fix the tag addresses that exhibit this behavior.

Read request for tag <tag address> on device <device name> failed due to a framing error.

Error Type:
Warning

Possible Cause:

1. A read request for the specified tag failed due to an incorrect request service code.
2. A read request for the specified tag failed because more or fewer bytes than expected were received.

Solution:

1. If this error occurs frequently, there may be an issue with the cabling or the device itself.
2. Increase the request attempts to allow more opportunities to recover from this error.

Unable to read tag <tag address> on device <device name>. [CIP Error=<code>, Ext. Error=<code>].

Error Type:
Warning

Possible Cause:
The device returned an error within the CIP portion of the Ethernet/IP packet during a read request for the specified tag.

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

See Also:
CIP Error Codes

Unable to read tag <tag address> on device <device name>. Address exceeds current CIP connection size.

Error Type:
Warning

Possible Cause:
The tag address size resulted in a request frame that exceeds the protocol limits.

Solution:
Increase the CIP Connection Size to a value that will accommodate the tag's request frame.

Unable to read tag <tag address> on device <device name>. Controller tag data type <type> unknown. Tag deactivated.

Error Type:
Warning

Possible Cause:
A read request for the specified tag failed because the Variable's data type is not currently supported.

Solution:
Remove references to this Variable. In response to this error, the tag will be deactivated; thus, it will not be processed again.

Unable to read tag <tag address> on device <device name>. Data type <type> is illegal for this tag.

Error Type:
Warning

Possible Cause:
A read request for the specified tag failed because the client's tag data type is illegal for the given Variable.

Solution:
Change the tag's data type to one that is supported.

See Also:
Data Type Coercion

Unable to read tag <tag address> on device <device name>. Data type <type> not supported. Tag deactivated.

Error Type:
Warning

Possible Cause:
A read request for the specified tag failed because the client's tag data type is not supported.

Solution:
Change the tag's data type to one that is supported. In response to this error, the tag will be deactivated; thus, it will not be processed again.

Unable to read tag <tag address> on device <device name>. Tag does not support multi-element arrays. Tag deactivated.

Error Type:
Warning

Possible Cause:
A read request for the specified tag failed because the driver does not support multi-element array access to the given variable.

Solution:
Change the tag's data type or address to one that is supported. In response to this error, the tag is deactivated and is not processed again.

Read Errors (Blocking)

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

Device <device name> returned more data than expected while reading <count> element(s) starting at <tag address>. Verify the address includes an element offset and all dimensions in that offset.
Read request for <count> element(s) starting at <tag address> on device <device name> failed due to a framing error.
Unable to read <count> element(s) starting at <tag address> on device <device name>. [CIP Error=<code>, Ext. Error=<code>].
Unable to read <count> element(s) starting at <tag address> on device <device name>. Address exceeds current CIP connection size.
Unable to read <count> element(s) starting at <tag address> on device <device name>. Block does not support multi-element arrays. Block deactivated.
Unable to read <count> element(s) starting at <tag address> on device <device name>. Controller Tag data type <type> unknown. Block deactivated.
Unable to read <count> element(s) starting at <tag address> on device <device name>. Data type <type> is illegal for this block.
Unable to read <count> element(s) starting at <tag address> on device <device name>. Data type <type> not supported. Block deactivated.

Device <device name> returned more data than expected while reading <count> element(s) starting at <tag address>. Verify the address includes an element offset and all dimensions in that offset.

Error Type:
Warning

Possible Cause:
The tag address references an Array Element Variable, but either not all dimensions were specified in the element offset or no element offset was specified at all. For example, the Array Element Variable "MyArray" is defined as ARRAY[0..9] OF INT. Creating a tag address like MyArray@Int may cause this error to occur because the first dimension is not specified. In this example, a read of the tag would result in 10 INTs returned versus the 1 INT that is expected.

Solution:

Add the fully-qualified element offset to the tag address. In the example above, the correct tag address could be MyArray[4].

Note:
This error message is only a warning. Reads will succeed, but will be less efficient because of the extra overhead in the read response. As such, it is recommended that users fix the tag addresses that exhibit this behavior.

Read request for <count> element(s) starting at <tag address> on device <device name> failed due to a framing error.

Error Type:
Warning

Possible Cause:

1. The specified tag address and count failed due to an incorrect request service code.
2. The specified tag address and count failed because more or fewer bytes than expected were received.

Solution:
If this error occurs frequently, there may be an issue with the cabling or the device itself. Increasing the request attempts allows more opportunities to recover from this error. In response to this error, <count> elements of the block are deactivated and it is not processed again.

Unable to read <count> element(s) starting at <tag address> on device <device name>. [CIP Error=<code>, Ext. Error=<code>].

Error Type:
Warning

Possible Cause:
The device returned an error within the CIP portion of the Ethernet/IP packet during a read request for the specified tag.

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

See Also:
CIP Error Codes

Unable to read <count> element(s) starting at <tag address> on device <device name>. Address exceeds current CIP connection size.

Error Type:
Warning

Possible Cause:
The tag address size resulted in a request frame that exceeds the protocol limits.

Solution:
Increase the CIP Connection Size to a value that will accommodate the tag's request frame.

Unable to read <count> element(s) starting at <tag address> on device <device name>. Block does not support multi-element arrays. Block deactivated.

Error Type:
Warning

Possible Cause:
A read request for the specified tags to the specified tag address and count failed because the driver does not support multi-element array access to the given Variable.

Solution:
Change the data type or address for tags within this block to one that is supported. In response to this error, <count> elements of the block will be deactivated; thus, it will not be processed again.

Unable to read <count> element(s) starting at <tag address> on device <device name>. Controller Tag data type <type> unknown. Block deactivated.

Error Type:
Warning

Possible Cause:
A read request for the specified tags to the specified tag address and count failed because the Variable's data type is not currently supported.

Solution:
Remove references to this Variable. In response to this error, <count> elements of the block will be deactivated; thus, it will not be processed again.

Unable to read <count> element(s) starting at <tag address> on device <device name>. Data type <type> is illegal for this block.

Error Type:
Warning

Possible Cause:
A read request for the specified tags to the specified tag address and count failed because the client's tag data type is illegal for the given Variable.

Solution:
Change the data type for tags within this block to one that is supported.

See Also:
Data Type Coercion

Unable to read <count> element(s) starting at <tag address> on device <device name>. Data type <type> not supported. Block deactivated.

Error Type:
Warning

Possible Cause:
A read request for the specified tags to the specified tag address and count failed because the client's tag data type is not supported.

Solution:
Change the data type for tags within this block to one that is supported. In response to this error, <count> elements of the block will be deactivated; thus, it will not be processed again.

Write Errors

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>.
Unable to write to tag <tag address> on device <device name>. [CIP Error=<code>, Ext. Error=<code>].
Unable to write to tag <tag address> on device <device name>. Address exceeds current CIP connection size.
Unable to write to tag <tag address> on device <device name>. Controller tag data type <type> unknown.
Unable to write to tag <tag address> on device <device name>. Data type <type> is illegal for this tag.
Unable to write to tag <tag address> on device <device name>. Data type <type> not supported.
Unable to write to tag <tag address> on device <device name>. Tag does not support multi-element arrays.

Write request for tag <tag address> on device <device name> failed due to a framing error.

Unable to write to <tag address> on device <device name>.

Error Type:
Warning

Possible Cause:

1. The Ethernet connection between the device and the host PC is broken.
2. The communication parameters for the Ethernet connection are incorrect.
3. The named device may have been assigned an incorrect IP address.

Solution:

1. Verify the cabling between the PC and the device.
2. Verify that the correct port has been specified for the named device.
3. Verify that the IP address given to the named device matches that of the actual device.

Unable to write to tag <tag address> on device <device name>. [CIP Error=<code>, Ext. Error=<code>].

Error Type:
Warning

Possible Cause:
The device returned an error within the CIP portion of the Ethernet/IP packet during a write request for the specified tag.

Solution:
The solution depends on the error code(s) returned. Consult the **CIP Error Codes**.

Unable to write to tag <tag address> on device <device name>. Address exceeds current CIP connection size.

Error Type:
Warning

Possible Cause:
The tag address size resulted in a request frame that exceeds the protocol limits.

Solution:
Increase the CIP connection size to a value that can accommodate the tag's request frame.

Unable to write to tag <tag address> on device <device name>. Controller tag data type <type> unknown.

Error Type:
Warning

Possible Cause:
A write request for the specified tag failed because the Variable's data type is not currently supported.

Solution:
Remove references to this Variable.

Unable to write to tag <tag address> on device <device name>. Data type <type> is illegal for this tag.

Error Type:
Warning

Possible Cause:
A write request for the specified tag failed because the client's tag data type is illegal for the given Variable.

Solution:
Change the tag's data type to one that is supported.

See Also:
Data Type Coercion

Unable to write to tag <tag address> on device <device name>. Data type <type> not supported.

Error Type:
Warning

Possible Cause:
A write request for the specified tag failed because the client's tag data type is not supported.

Solution:
Change the tag's data type to one that is supported.

Unable to write to tag <tag address> on device <device name>. Tag does not support multi-element arrays.

Error Type:
Warning

Possible Cause:
A write request for the specified tag failed because the driver does not support multi-element array access to the given Variable.

Solution:
Change the tag's data type or address to one that is supported.

Write request for tag <tag address> on device <device name> failed due to a framing error.

Error Type:
Warning

Possible Cause:

1. A write request for the specified tag failed after so many retries due to an incorrect request service code.
2. A write request for the specified tag failed after so many retries because more or fewer bytes than expected were received.

Solution:
If this error occurs frequently, there may be an issue with the cabling or device. Increasing the retry attempts allows more opportunities to recover from this error.

Error Codes

The following sections define error codes that may be encountered in the server's Event Log. For more information on a specific error code type, select a link from the list below.

Encapsulation Error Codes
CIP Error Codes

Encapsulation Error Codes

The following error codes are in hexadecimal.

Status (Hex)	Description
0001	Sender issued an invalid or unsupported encapsulation command.
0002	Insufficient memory resources available in the receiver to handle the command.
0003	Poorly formed or incomplete information in the data portion of the encapsulation message.
0004 - 0063	Reserved
0064	Originator used an invalid session handle when sending an encapsulation message to the target.
0065	Invalid length in header.
0066 - 0068	Reserved
0069	Requested protocol version is not supported.
006A - FFFF	Reserved

CIP Error Codes

The following error codes are in hexadecimal.

Status (Hex)	Description
01	Connection-related service failed along the connection path. *See Also:* ***0x01 Extended Error Codes***
02	Resources needed for the object to perform the requested service were unavailable.
03	Invalid parameter value.
04	Path segment error. Tag does not exist in the device.
05	Path destination unknown. Structure member does not exist or array element is out of range.
06	Partial transfer; only part of the expected data was transferred.
07	Loss of connection.
08	Service not supported. The requested service was not implemented or was not defined for this class or instance.
09	Invalid attribute value.
0A	Attribute list error.
0B	Object is already in the mode/state requested by the service.
0C	Object cannot perform the requested service in its current mode/state. Project change may be in progress. *See Also:* ***0x0C Extended Error Codes***
0D	Requested instance of object to be created already exists.
0E	A request to modify a non-editable attribute was received.
0F	A permission / privilege check failed.
10	The device's current mode/state prohibits the execution of the requested service.
11	Reply data too large. The data to be transmitted in the response buffer is larger than the allocated response buffer.
12	The service specified an operation that would fragment a primitive data value.
13	Not enough data. The service did not supply enough data to perform the specified operation.
14	Attribute not supported.
15	Too much data. The service supplied more data than expected.
16	The object specified does not exist in the device.
17	The fragmentation sequence for this service is not currently active for this data.
18	The attribute data of this object was not saved prior to the requested service.

Status (Hex)	Description
19	The attribute data of this object was not saved due to a failure during the attempt.
1A	Routing failure; request packet too large.
1B	Routing failure; response packet too large.
1C	Missing attribute in list entry data.
1D	Invalid attribute value list.
1E	Embedded service error. One or more services returned an error within a multiple-service packet service.
1F	Vendor-specific error. Consult vendor documentation. *See Also:* ***0x1F Extended Error Codes***
20	Invalid parameter. Parameter does not meet the requirements of the CIP specification or Omron specification. *See Also:* ***0x20 Extended Error Codes***
21	An attempt was made to write to a write-once medium that has already been written.
22	Invalid reply received. Reply service code does not match the request service code or reply message is shorter than the minimum expected reply size.
23	The message received is larger than the receiving buffer can handle.
24	The format of the received message is not supported by the server.
25	The key segment included as the first segment in the path does not match the destination module.
26	The size of the path sent with the service request is not large enough to allow the request to be routed to an object or too much routing data was included.
27	Unexpected attribute in list.
28	The member ID specified in the request does not exist in the specified class, instance, or attribute.
29	A request to modify a non-modifiable member was received.
2A	DeviceNet-specific error.
2B	A CIP to Modbus translator received an unknown Modbus exception code.
2C	A request to read a non-readable attribute was received.
2D	A requested object instance cannot be deleted.
2E	The object supports the service, but not for the designated application path (for example, attribute).
2F - CF	Reserved by CIP.
D0 - FF	Object class specific errors.

0x01 Extended Error Codes

The following error codes are in hexadecimal.

Extended Status (Hex)	Description
0100	Connection in use or duplicate Forward Open request. Originator is trying to make a connection to a target with an established connection.
0101 - 0102	Reserved by CIP.
0103	Transport class and trigger combination specified is not supported by the target application.
0104	Reserved by CIP.
0105	See CIP Safety Specification.
0106	Ownership conflict. The connection cannot be established because another connection has exclusively allocated some of the resources required for this connection.
0107	Target connection not found. Typically returned in response to Forward Close request when the connection to be closed is not found at the target node.
0108	Invalid network connection parameter. Connection type, priority, or fixed/variable is not supported by the device.
0109	Invalid connection size. Target does not support the specified connection size.
010A - 010F	Reserved by CIP.
0110	Target for connection not configured.
0111	RPI not supported. Returned if device cannot support the request (O->T or T->O RPI) or the connection timeout multiplier produces a timeout value that is not supported by the device.
0112	RPI value(s) not acceptable. Returned if the RPI value(s) in the Forward Open request are outside the range required by the application in the target device

Extended Status (Hex)	Description
	or the target is producing at a different interval.
0113	Out of connections. The maximum number of connections supported by the Connection Manager has been reached.
0114	Vendor ID or product code specified in the electronic key logical segment does not match the product code or vendor ID of the device.
0115	Device type specified in the electronic key logical segment does not match the device type of the device.
0116	The major and minor revision specified in the electronic key logical segment do not match the revision of the device.
0117	The produced or consumed application path specified in the connection path does not correspond to a valid produced or consumed application path within the target application.
0118	An application path specified for the configuration data does not correspond to a configuration application or is inconsistent with the consumed or produced application path.
0119	No non-listen only-connection types currently open. Returned when an attempt is made to establish a listen-only type to a target which has no non-listen-only connection already established.
011A	The maximum number of connections supported by this instance of the target object has been exceeded.
011B	The Production Inhibit Time is greater than the T->O RPI.
011C	Transport class requested in the transport type/trigger parameter is not supported.
011D	Production trigger requested in the transport type/trigger parameter is not supported.
011E	Direction requested in the transport type/trigger parameter is not supported.
011F	O->T fixed/variable flag is not supported.
0120	T->O fixed/variable flag is not supported.
0121	O->T Priority code is not supported.
0122	T->O Priority code is not supported.
0123	O->T Connection type is not supported.
0124	T->O Connection type is not supported.
0125	O->T Redundant owner flag is not supported.
0126	Data segment in the connection path parameter does not contain an acceptable number of 16 bit words for the configuration application path requested.
0127	Size of the consuming object declared in the Forward Open request and available on the target does not match the connection size declared in the O->T network connection parameter.
0128	Size of the producing object declared in the Forward Open request and available on the target does not match the connection size declared in the T->O network connection parameter.
0129	Configuration application path specified in the connection path does not correspond to a valid configuration application path within the target application.
012A	Consumed application path specified in the connection path does not correspond to a valid consumed application path within the target application.
012B	Produced application path specified in the connection path does not correspond to a valid produced application path within the target application.
012C	Originator attempted to connect to a configuration tag name not in the list of tags defined in the target.
012D	Originator attempted to connect to a consuming tag name not in the list of tags defined in the target.
012E	Originator attempted to connect to a producing tag name not in the list of tags defined in the target.
012F	Combination of configuration, consume, and/or produce application paths specified in the connection path are inconsistent with each other.
0130	Information in the data segment is not consistent with the format of the consumed data.
0131	Information in the data segment is not consistent with the format of the

Extended Status (Hex)	Description
	produced data.
0132	Null Forward Open request is not supported by target.
0133	Connection timeout multiplier (inactivity watchdog) specified is reserved or produces a timeout value too large for the device.
0134 - 0202	Reserved by CIP.
0203	Connection timed out.
0204	Unconnected request timeout. This may be the result of congestion at the destination node or a node not being powered up or present.
0205	Connection tick time and connection timeout combination in unconnected request is not supported by an intermediate node.
0206	Message too large for unconnected_send service.
0207	Unconnected message acknowledge was received, but a data response message was not received.
0208 - 0300	Reserved by CIP.
0301	Insufficient connection buffer memory is available.
0302	Producer along path cannot allocate sufficient bandwidth for the connection on its link.
0303	No consumed connection ID filter available.
0304	Device unable to send scheduled priority data.
0305	Connection scheduling information in the originator device is not consistent with the connection scheduling information on the target network.
0306	Connection scheduling information in the originator device cannot be validated on the target network.
0307 - 0310	Reserved by CIP.
0311	A port specified in a port segment is not available or does not exist.
0312	Link address specified in a port segment is not valid for the target network type.
0313 - 0314	Reserved by CIP.
0315	Invalid segment type or segment value in the connection path.
0316	The connection path in the Forward Close service does not match the connection path in the connection being closed.
0317	Scheduled network segment is not present or the value within the segment is invalid.
0318	Link address to self (loopback) is invalid.
0319	Secondary system in dual-chassis redundant system is unable to duplicate connection request made to primary system.
031A	Request for a module connection has been refused because part of the corresponding data is already included in a rack connection.
031B	Request for a rack connection has been refused because part of the corresponding data is already included in a module connection.
031C	Miscellaneous connection-related error occurred.
031D	Redundant connection mismatch.
031E	The configured number of consumers for a producing application is already reached.
031F	No consumers configured for a producing application to use.
0320 - 07FF	Vendor-specific error.
0800	Network link in path to module is offline.
0801 - 080F	See CIP Safety Specification.
0810	Target application does not have valid data to produce for the requested connection.
0811	Originator application does not have valid data to produce for the requested connection.
0812	Node address has changed since the network was scheduled.
0813	A multicast connection has been requested between a producer and a consumer that are on different subnets and the producer is not configured for off-subnet multicast.
0814	Information in the data segment indicates that the format of the produced and/or consumed data is not valid.

Extended Status (Hex)	Description
0815 - FCFF	Reserved by CIP.

0x0C Extended Error Codes

The following error codes are in hexadecimal.

Extended Status (Hex)	Description
8010	A download is in progress.
8011	There is an error in tag memory.

Note: For unlisted error codes, refer to the Omron documentation.

0x1F Extended Error Codes

The following error codes are in hexadecimal.

Extended Status (Hex)	Description
0101	Could be one of the following errors: 1. The combined size of the variable type and read address is incorrect. 2. The variable type specification is incorrect. 3. The read start address exceeds the range of the variable area. 4. The read end address exceeds the range of the variable area. 5. There are too many elements.
0102	Could be one of the following errors: 1. The number of elements does not match the size of the write data. 2. The variable type specification is not correct. 3. A Read Only area is included in the write area.
0104	A variable type is out of range.
8001	An internal error occurred.
800D	There is an error in the registered tag information.
8014	An internal error occurred.
8016	A variable is not correctly registered.

Note: For unlisted error codes, refer to the Omron documentation.

0x20 Extended Error Codes

The following error code is in hexadecimal.

Extended Status (Hex)	Description
8017	More than one element was specified for a variable that does not have elements.
8018	Zero elements or data that exceeded the range of the array was specified for an array.
8022	The data type specified in the request service data does not agree with the tag information. The AddInfo Length in the request service data is not 0.
8028	Value is out of range.

Note: For unlisted error codes, refer to the Omron documentation.

Glossary

Term	Definition
Link Address	Unique identifier for an Omron unit (such as Unit Address, IP address, and so forth).
Packet	Stream of data bytes on the wire representing the request(s) being made. Packets are limited in size.
Port Number	Specifies a way out of the Omron unit in question (such as a channel).
Routing	Utilizing one or more Omron NJ racks to hop to another Omron NJ rack.

*For more information on tag division, refer to **Performance Statistics and Tuning** and **Optimizing Communications**.

Variable Tag-Based Addressing

Term	Definition
Array	Client/Server Array Tag whose address has an Array Element specified. For example, ARRAYTAG [0] {5}.
Array Element	Element within a Variable Array. For client/server access, the element must be a basic. For example, ARRAYTAG [0].
Array Variable	Multi-dimensional array (1, 2 or 3 dimensions possible) support within Sysmac Studio for Omron NJ-platform controllers. All basic data types support Variable Arrays. Not all basic structure data types support Variable Arrays.
Basic Data Type	A pre-defined, non-structured data type. Example: SINT and DINT.
Basic Tag	A Variable Tag defined with a basic data type.
Client	An HMI/SCADA or data bridging software package utilizing OPC,DDE, or proprietary client/server protocol to interface with the server.
Client/Server Data Type	Data type for tags defined statically in the server or dynamically in a client. Supported data types in the server are listed in Data Type Descriptions. Supported data types in the client depends on the client in use.
Client/Server Tag	Tag defined statically in the server or dynamically in a client. These tags are different entities than Variable Tags. A Variable Tag's tag name becomes a Client/Server Tag's tag address when referencing a Variable Tag.
Client/Server Array	Row x column data presentation format supported by the server and by some clients. Not all clients support arrays.
Variable Data Type	A data type defined in Sysmac Studio for Omron NJ-platform controllers.
Variable Tag	Tag defined in Sysmac Studio for Omron NJ-platform controllers.
Server	The OPC/DDE/proprietary server utilizing this Omron NJ Ethernet Driver.
Structure Data Type	A complex data type (pre-defined or user-defined) that consists of members whose data types are basic or structured in nature.
Structure Tag	A Variable Tag defined with a Structure Data Type.

Index

0

A

B

C

D

E

F

G

I

L

M

N

O

P

R

S

T

U

W