Aromat Ethernet Driver

Table of Contents

Aromat Ethernet Driver

Help version 1.016

CONTENTS

Overview

The Aromat Ethernet Driver provides a reliable way to connect Aromat Ethernet devices to OPC client applications; including HMI, SCADA, Historian, MES, ERP, and countless custom applications. It is intended for use with Aromat Matsushita/NAIS FP devices with ET-LAN units.

Setup

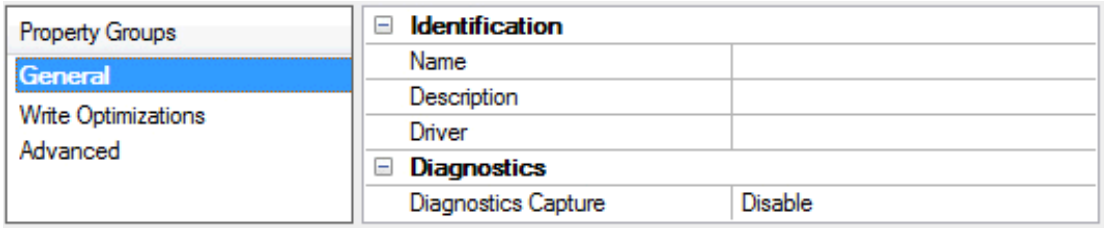
Supported Devices

Any Aromat Matsushita/NAIS FP series PLC with optional ET-LAN Ethernet unit.

Note: This driver does not support multi-level networks where messages are relayed from one PLC to another. It is possible to communicate with PLCs on other network levels if messages are relayed through a router.

Channel Properties - General

This server supports the use of simultaneous multiple communications drivers. Each protocol or driver used in a server project is called a channel. A server project may consist of many channels with the same communications driver or with unique communications drivers. A channel acts as the basic building block of an OPC link. This group is used to specify general channel properties, such as the identification attributes and operating mode.



Identification

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

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

Driver: Selected protocol / driver for this channel. This property specifies the device driver that was selected during channel creation. It is a disabled setting in the channel properties.

Note: With the server's online full-time operation, these properties can be changed at any time. This includes changing the channel name to prevent clients from registering data with the server. If a client has already acquired an item from the server before the channel name is changed, the items are unaffected. If, after the channel name has been changed, the client application releases the item and attempts to re-acquire using the old channel name, the item is not accepted. With this in mind, changes to the properties should not be made once a large client application has been developed. Utilize the User Manager to prevent operators from changing properties and restrict access rights to server features.

Diagnostics

Diagnostics Capture: When enabled, this option makes the channel's diagnostic information available to OPC applications. Because the server's diagnostic features require a minimal amount of overhead

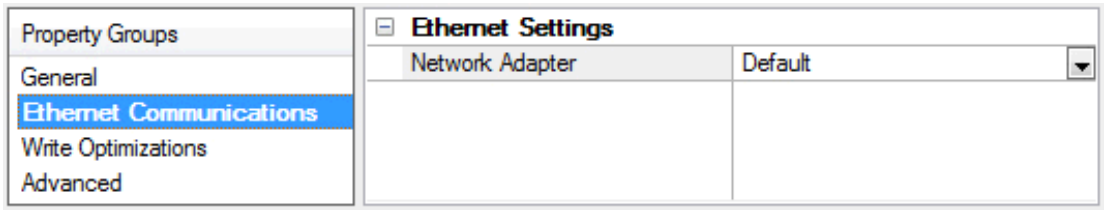
processing, it is recommended that they be utilized when needed and disabled when not. The default is disabled.

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

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

Channel Properties - Ethernet Communications

Ethernet Communication can be used to communicate with devices.

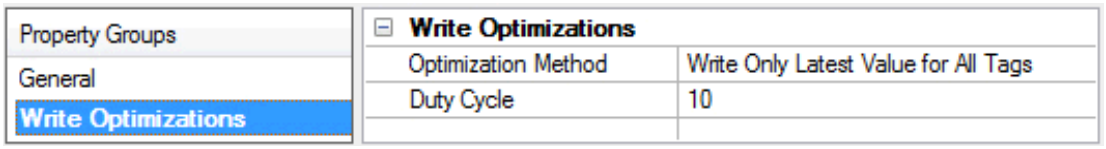


Ethernet Settings

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

Channel Properties - Write Optimizations

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



Write Optimizations

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

- **Write All Values for All Tags**: This option forces the server to attempt to write every value to the controller. In this mode, the server continues to gather write requests and add them to the server's internal write queue. The server processes the write queue and attempts to empty it by writing data to the device as quickly as possible. This mode ensures that everything written from the client applications is sent to the target device. This mode should be selected if the write operation order or the write item's content must uniquely be seen at the target device.
- **Write Only Latest Value for Non-Boolean Tags**: Many consecutive writes to the same value can accumulate in the write queue due to the time required to actually send the data to the device. If the server updates a write value that has already been placed in the write queue, far fewer writes are needed to reach the same final output value. In this way, no extra writes accumulate in the server's queue. When the user stops moving the slide switch, the value in the device is at the correct value at virtually the same time. As the mode states, any value that is not a Boolean value is updated in the server's internal write queue and sent to the device at the next possible opportunity. This can greatly

improve the application performance.

Note: This option does not attempt to optimize writes to Boolean values. It allows users to optimize the operation of HMI data without causing problems with Boolean operations, such as a momentary push button.

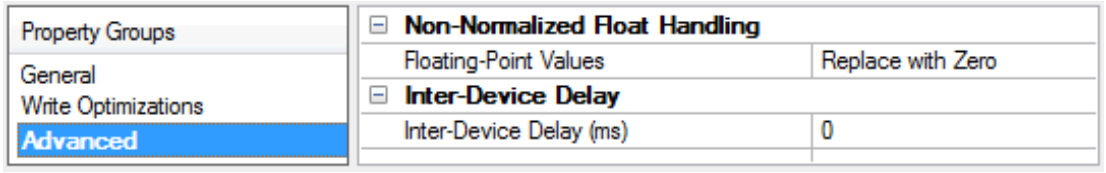
- **Write Only Latest Value for All Tags**: This option takes the theory behind the second optimization mode and applies it to all tags. It is especially useful if the application only needs to send the latest value to the device. This mode optimizes all writes by updating the tags currently in the write queue before they are sent. This is the default mode.

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

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

Channel Properties - Advanced

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



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

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

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

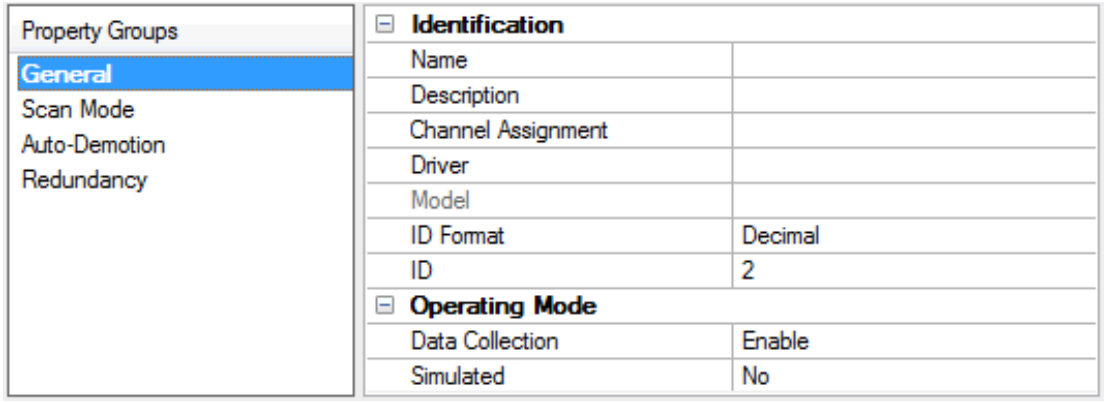
For more information on the floating point values, refer to "How To ... Work with Non-Normalized Floating Point Values" in the server help.

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

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

Device Properties - General

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



Identification

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

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

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

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

Driver: Selected protocol driver for this device. This property specifies the driver selected during channel creation. It is disabled in the channel properties.

Model: This property specifies the specific type of device that is associated with this ID. The contents of the drop-down menu depends on the type of communications driver being used. Models that are not supported by a driver are disabled. If the communications driver supports multiple device models, the model selection can only be changed when there are no client applications connected to the device.

Note: If the communication driver supports multiple models, users should try to match the model selection to the physical device. If the device is not represented in the drop-down menu, select a model that conforms closest to the target device. Some drivers support a model selection called "Open," which allows users to communicate without knowing the specific details of the target device. For more information, refer to the driver help documentation.

ID: This property specifies the device's driver-specific station or node. The type of ID entered depends on the communications driver being used. For many communication drivers, the ID is a numeric value. Drivers that support a Numeric ID provide users with the option to enter a numeric value whose format can be

changed to suit the needs of the application or the characteristics of the selected communications driver. The ID format can be Decimal, Octal, and Hexadecimal.

Note: If the driver is Ethernet-based or supports an unconventional station or node name, the device's TCP/IP address may be used as the device ID. TCP/IP addresses consist of four values that are separated by periods, with each value in the range of 0 to 255. Some device IDs are string based. There may be additional properties to configure within the ID field, depending on the driver. For more information, refer to the driver's help documentation.

Operating Mode

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

Simulated: This option places the device into Simulation Mode. In this mode, the driver does not attempt to communicate with the physical device, but the server continues to return valid OPC data. Simulated stops physical communications with the device, but allows OPC data to be returned to the OPC client as valid data. While in Simulation Mode, the server treats all device data as reflective: whatever is written to the simulated device is read back and each OPC item is treated individually. The item's memory map is based on the group Update Rate. The data is not saved if the server removes the item (such as when the server is reinitialized). The default is No.

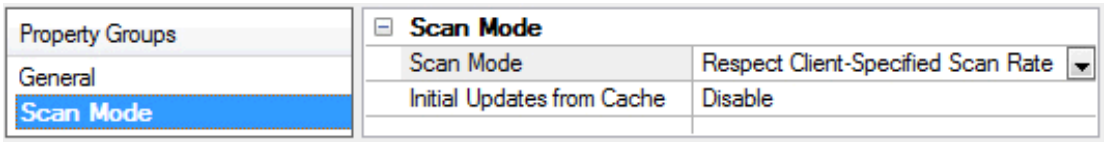
Notes:

1. This System tag (_Simulated) is read only and cannot be written to for runtime protection. The System tag allows this property to be monitored from the client.
2. In Simulation mode, the item's memory map is based on client update rate(s) (Group Update Rate for OPC clients or Scan Rate for native and DDE interfaces). This means that two clients that reference the same item with different update rates return different data.

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

Device Properties - Scan Mode

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



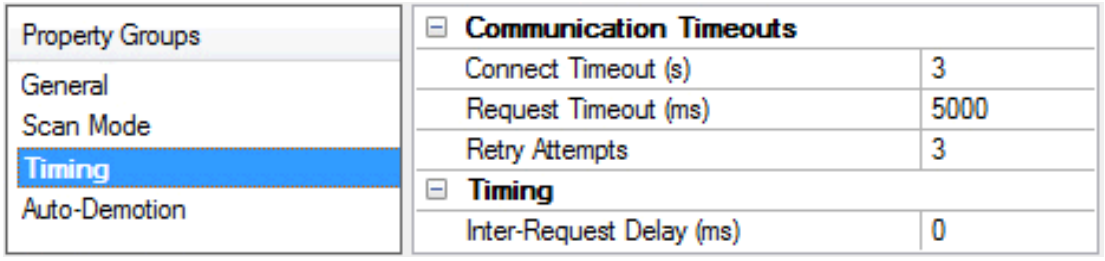
Scan Mode: specifies how tags in the device are scanned for updates sent to subscribed clients. Descriptions of the options are:

- **Respect Client-Specified Scan Rate**: This mode uses the scan rate requested by the client.
- **Request Data No Faster than Scan Rate**: This mode specifies the maximum scan rate to be used. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
  Note: When the server has an active client and items for the device and the scan rate value is increased, the changes take effect immediately. When the scan rate value is decreased, the changes do not take effect until all client applications have been disconnected.
- **Request All Data at Scan Rate**: This mode forces tags to be scanned at the specified rate for subscribed clients. The valid range is 10 to 99999990 milliseconds. The default is 1000 milliseconds.
- **Do Not Scan, Demand Poll Only**: This mode does not periodically poll tags that belong to the device nor perform a read to get an item's initial value once it becomes active. It is the client's responsibility to poll for updates, either by writing to the _DemandPoll tag or by issuing explicit device reads for individual items. *For more information, refer to "Device Demand Poll" in server help*.
- **Respect Tag-Specified Scan Rate**: This mode forces static tags to be scanned at the rate specified in their static configuration tag properties. Dynamic tags are scanned at the client-specified scan rate.

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

Device Properties - Timing

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



Communications Timeouts

Connect Timeout: This property (which is used primarily by Ethernet based drivers) controls the amount of time required to establish a socket connection to a remote device. The device's connection time often takes longer than normal communications requests to that same device. The valid range is 1 to 30 seconds. The default is typically 3 seconds, but can vary depending on the driver's specific nature. If this setting is not supported by the driver, it is disabled.
Note: Due to the nature of UDP connections, the connection timeout setting is not applicable when communicating via UDP.

Request Timeout: This property specifies an interval used by all drivers to determine how long the driver waits for a response from the target device to complete. The valid range is 50 to 9,999,999 milliseconds (167.6667 minutes). The default is usually 1000 milliseconds, but can vary depending on the driver. The

default timeout for most serial drivers is based on a baud rate of 9600 baud or better. When using a driver at lower baud rates, increase the timeout to compensate for the increased time required to acquire data.

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

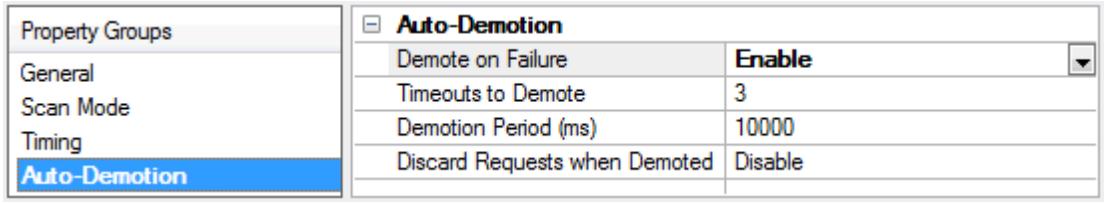
Timing

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

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

Device Properties - Auto-Demotion

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



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

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

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

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

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

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

Device Properties - Communications Properties

Protocol: Specify whether the driver should connect to the ET-LAN unit using the User Datagram Protocol (UDP) or the Transfer Control Protocol (TCP). The driver requires Winsock V1.1 or higher.

Open Method: When the TCP protocol is used, this property will specify the method that the ET-LAN unit is configured to use in order to process connection open requests. Methods include Active Open, Unpassive Open and Full Passive Open. If configured to use the Active Open method, the ET-LAN unit will actively attempt to initiate the connection. If configured to use the Unpassive Open method, the ET-LAN unit will wait passively for another node (which may use any IP and port number) to initiate the connection. If configured to use the Full Passive Open method, the ET-LAN unit will wait passively for a node (which uses a specified IP and port) to initiate the connection. Each connection that could possibly be made to the ET-LAN unit must be individually configured in the unit. For more information, refer to **Connection Types**.
Note: Although the Active Open method cannot be used to connect with this driver, it may be used for PLC-to-PLC connections.
Important: There is a known operating system issue that prevents the driver from reusing a given source port and IP with TCP/IP for 4 or more minutes after a server shutdown. Because of this, it is strongly recommended that the TCP/Unpassive or UDP connection method be used intead of the TCP/Full Passive method.

Source Port: Specify which port the driver should use to send and receive messages. It only applies to UDP and TCP/Full Open. If TCP/Unpassive mode is used, the operating system will automatically select an unused source port number. It is recommended that the chosen port number be greater than 1024. The default setting is 1025. Certain restrictions apply. *For more information, refer to **Port Assignment**.*

Destination Port: The Destination Port Number specifies what port the ET-LAN is configured to send and receive messages. The default port number is 1025. It is recommended that port numbers greater than 1024 be used. Certain restrictions apply. *For more information, refer to **Port Assignment**.*

See Also:ET-LAN Connection Properties

Device Properties - Station Numbers

Source Station: Specify the station number that will identify the driver's connection point. Each node on the network must be assigned a unique station number. This includes the connection points in the driver (one per device object). Valid numbers range from 1 to 64, where 1 is the default. *For more information, refer to **ET-LAN Connection Properties.***

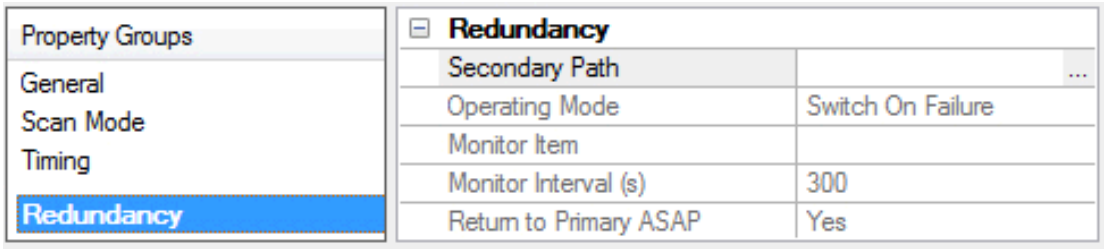
Destination Station: Specify the target device's station number. Valid numbers range from 1 to 64, where 1 is the default.

See Also: Setup

Device Properties - Request Size

Request Size: Specify the number of bytes that may be requested from a device at one time. To refine the performance of this driver, the request size may be configured to one of the following settings: 32, 64, 128, 256 and 512 bytes. The default value is 64 bytes. If a large number of closely spaced addresses will be read, it is most likely advantageous to specify a larger request size. If a few addresses that are widely separated will be read, then it is most likely advantageous to use a smaller request size.

Device Properties - Redundancy



Redundancy is available with the Media-Level Redundancy Plug-In.

Consult the website, a sales representative, or the user manual for more information.

ET-LAN Connection Properties

Overview

Each ET-LAN unit can maintain up to eight connections, which can be made to other PLCs, routers or to this driver. Each connection must be described ahead of time in the PLC's shared memory area. Connection descriptions include the following:

- Protocol
- Open Processing Method
- Source and Destination IP
- Source and Destination Port Numbers
- Source and Destination Station Numbers
- Routing Information

Note: The shared memory area is configured using a ladder program.

Connection Types

The ET-LAN unit supports communication using the User Datagram Protocol (UDP) and the Transfer Control Protocol (TCP).

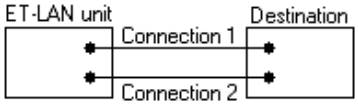
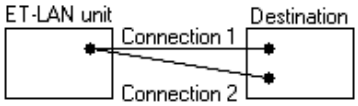
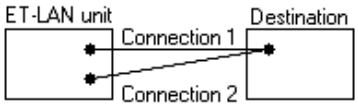
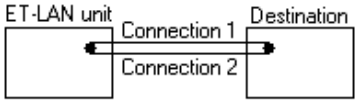
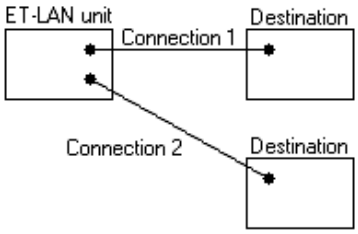
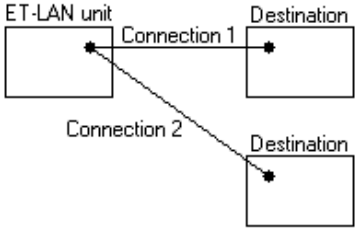
UDP connections are initiated by the driver. The IP and port used by the driver must be specified in the PLC's shared memory area. When programming the device to use UDP, it may be necessary to cycle the power for changes to take effect.

Requests to make connections using TCP must be processed using one of three methods: Active Open, Unpassive Open and Full Passive Open. Descriptions are as follows:

- **TCP/Active Open:** Connections are initiated by the PLC and are not supported by this driver. Active Open connections can still be used for PLC-to-PLC communication.
- **TCP/Unpassive Open:** Connections are initiated by the driver. The IP and port used by the driver do not need to be specified in the PLC's shared memory area. This is the preferred connection method.
- **TCP/Full Passive Open:** Connections are initiated by the driver. The IP and port used by the driver must be specified in the PLC's shared memory area. There is a known operating system issue that prevents the driver from reusing a given source port and IP with TCP/IP for 4 or more minutes after a server shutdown. Because of this, it is strongly recommended that the TCP/Unpassive or UDP connection method be used instead.

Port Assignment

Port numbers are used to distinguish each of the various communication processes. It is recommended that port numbers be greater than 1024, since many of the lower port numbers are commonly reserved by computer operating systems. When multiple connections are made to an ET-LAN unit, there are further restrictions on the port numbers (as illustrated below). A similar table may be found in the ET-LAN documentation. The descriptions below explain how the configuration relates to a server project.

Combination			Availability	
Destination configuration	Port number settings (each circle is a unique port number)	Description	Communication method	
			TCP	UDP
ET-LAN is connected to one destination node		Destination is another PLC or two device objects using different ports on same channel or on different channels using same IP.	1	1
		Destination is another PLC or two device objects using different ports on same channel or on different channels using same IP.	1, 2	N/A
		Destination is another PLC or two device objects using the same port on same channel or on different channels using same IP.	1	N/A
		Destination is another PLC or two device objects using the same port on same channel or on different channels using same IP.	N/A	N/A
ET-LAN is connected to multiple destination nodes		Each destination could be another PLC or a device object. If two device objects, they are on different channels using different IP's.	1, 3	1, 3
		Each destination could be another PLC or a device object. If two device objects, they are on different channels using different IP's.	1, 2, 4	N/A

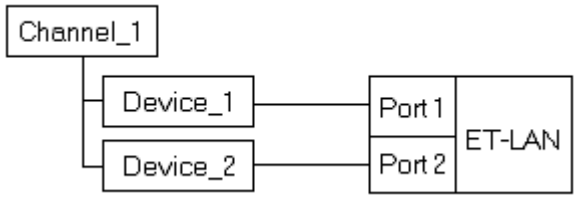
Notes

1. If the destination is two server device objects, they each must be assigned a unique station number.
2. This is not available with FP2 ET1 unit.

3. This configuration will also work if the channels use the same IP and each device uses a unique port. If channels use different IPs, then device objects may use the same port number.
4. A potential message routing problem exists here. If destinations are two server device objects, they must not use the same IP and port numbers.

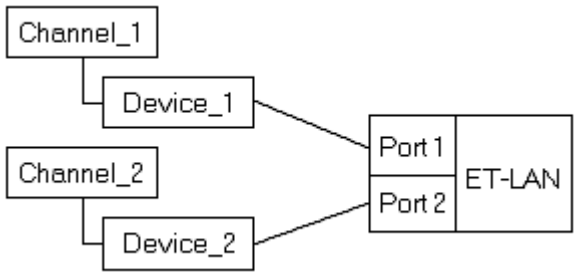
Recommended Configuration for Multiple Connections

There should generally be a one-to-one relationship between server device objects and PLCs. It is possible to configure multiple server device objects that connect with the same PLC, but that may result in a small increase in performance. To create multiple connections to a single PLC, it is recommended that each device object use a unique combination of IP (selected from channel network adaptor) and source port number illustrated below. For more information on how to create multiple source IPs, refer to **Multi- Homing**.



Multiple connections, single channel

Device_1 and Device_2 should use unique port and station numbers.



Multiple connections, multiple channels

If Channel_1 and Channel_2 use the same IP, then Device_1 and Device_2 should use unique port numbers. If the channels use different IPs, then the devices may use the same port number.

Both devices should use unique station numbers.

Station Numbers

Each device object that connects with the same PLC should have a unique source station number. Server device objects that connect with different PLCs may use the same source station number.

Configuring the Shared Memory Area

Each possible connection to the PLC must be described in the shared memory area. The shared memory area is configured using a ladder program such as the one illustrated below. For more information, refer to the ET-LAN documentation.

This program configures a single ET-LAN unit in slot 0 in order to connect to two external devices. The external devices may be two other PLCs, two server device objects or a combination. Each connection uses the UDP protocol. The PLC uses a unique port for each connection: 6004 and 6005. The first external device uses IP 192.168.111.27 (0xC0A86F1B), port 1025 and station number 10. The second external device uses IP 192.168.111.70 (0xC0A86F46), port 1026 and station number 11. A subnet mask of 255.255.255.255 (0xFFFFFE00) and a default router/gateway IP of 192.168.111.1 (0xC0A86F01) are specified, but are not required in this particular application.

Allocation of handshake area for internal relays:

```
| |
| |
| R9010 |
|----| |----[F150 READ, H 0, H 360, K 2, WR 0] |
| [F151 WRT, H 0, WR 4, K 2, H 368] |
| |
| |
```

Initialization processing:

```
| |
| |
| R9013 |
|----| |----[F1 DMV, H C0A86F83, DT 10] |
| [F0 MV, H 1, DT 12] |
| [F0 MV, K 1, DT 13] |
| [F151 WRT, H 0, DT 10, K 4, H 200] |
| |
| R9014 RD R4C |
|----| |--------|/|-----------------------------------------[ ]----|
| |
| |
```

Initialization of routing information:

```
| |
| |
| RD  |
|----| |----[F150 READ, H 0, H 2D0, K1, DT 300] |
| |
| R9013 |
|----| |----[F1 DMV, H FFFFFE00, DT 14] |
| [F1 DMV, H C0A86F01, DT 16] |
| [F151 WRT, H 0, DT 14, K 4, H 230] |
| |
| |
```

Initialization of connection 1 information:

```
| |
| |
| R9013 |
|----| |----[F0 MV, H 8000, DT 20] |
| [F0 MV, K 6004, DT 21] |
| [F0 MV, H C0A86F1B, DT 22] |
| [F0 MV, K 1025, DT 24] |
| [F0 MV, K 10, DT 25] |
| [F151 WRT, H 0, DT 20, K 6, H 250] |
| |
| RC R11 R50 |
|----| |--------|/|-----------------------------------------[ ]----|
| |
| |
```

Initialization of connection 2 information:

```
| |
| |
| R9013 |
|----| |----[F0 MV, H 8000, DT 30] |
| [F0 MV, K 6005, DT 31] |
| [F0 MV, H C0A86F46, DT 32] |
```

```
| [F0 MV, K 1026, DT 34] |
| [F0 MV, K 11, DT 35] |
| [F151 WRT, H 0, DT 30, K 6, H 260] |
| |
| RC R13 R52 |
|----| |--------|/|-----------------------------------------[ ]----|
| |
| |
```

RC is initialization complete signal
RD is initialization error signal
R11 is open error signal (connection 1)
R13 is open error signal (connection 2)
R4C is initialization request signal
R50 is open complete signal (connection 1)
R52 is open complete signal (connection 2)
R9010 is always ON relay
R9013 is initial ON relay
R9014 is initial OFF relay

Connection Information Shared Memory Area

The connection information shared memory area is allotted as follows:

Connection	Address Range
1	0x250-0x25F
2	0x260-0x26F
3	0x270-0x27F
4	0x280-0x28F
5	0x290-0x29F
6	0x2A0-0x2AF
7	0x2B0-0x2BF
8	0x2C0-0x2CF

Offset	Description
0	Protocol and Open Method: 0x8000 UDP. 0x0300 TCP/Full Passive Open. 0x0200 TCP/Unpassive Open. 0x0000 TCP/Active Open.*
1	Source node port number.
2	Partner node IP (low word).
3	Partner node IP (high word).
4	Partner node port number.
5	Partner node station number.
6	N/A**

Offset	Description
7	N/A**
8	N/A**
9	Reserved.
A	Reserved.
B	Reserved.
C	Reserved.
D	N/A***
E	Reserved.
F	N/A****

*This is not supported by the driver.
**Partner node physical address.
***Receive request data size.
****Transmission request data size.

Multi-Homing

Some applications require a unique IP address be associated with each channel. In these cases, the host computer must be multi-homed. A computer is multi-homed when it is configured to have more than one IP address. This may be accomplished by installing multiple Network Adapter Cards (NICs) in the computer or by assigning multiple IP addresses to a single NIC.

Adding IP Addresses to a Single NIC on Windows NT

1. Double-click the **My Computer** icon. Select **Control Panel**.
2. Click **Network** and then select the **Protocols** tab.
3. Select **TCP/IP Protocol**.
4. Click **Properties** and then select the **IP Address** tab.
5. Click **Advanced**.
6. Click **Add**.
7. Enter the additional **IP address** and **subnet mask**.
8. Click **OK**.

Notes:

- Multi-homing is not supported under Windows 95 or 98.
- Windows NT will allow up to 5 IP addresses to be added for each NIC via the control panel. If more IP addresses are necessary, they can be added to the registry manually by browsing under **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services**. Next, select the service associated with the adapter card in question. Under the service, go to the **Parameters\Tcpip** subkey. Add the IP addresses to IP Address. Edit Subnet Mask and add an entry for each new IP address.
- The system may need to be restarted before newly added IPs can be used.

- There will be additional operating system overhead when running on a multi-homed system. Unless a very fast device is being used, this overhead should not entirely cancel out the performance gain achieved from distributing the communications load over multiple channels.

Data Types Description

Data Type	Description
Boolean	Single bit
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
DWord	Unsigned 32 bit value bit 0 is the low bit bit 31 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
BCD	Two byte packed BCD Value range is 0-9999. Behavior is undefined for values beyond this range.
LBCD	Four byte packed BCD Value range is 0-99999999. Behavior is undefined for values beyond this range.
Float	32 bit Floating point value The driver interprets two consecutive registers as a Floating-point value by making the second register the high word and the first register the low word.

Address Descriptions

Address specifications vary depending on the model in use. The address ranges shown below may exceed the range available for a particular device. If an address is requested that is not supported by the device, the Aromat Matsushita/NAIS driver will mark the requested data item in error.

Note: The default data types are shown in **bold**.

Address Types

Type	Valid Tag Prefixes	Valid Data Types
I/Os**	X, Y	**Boolean**
I/Os***	WX, WY	**Word**, DWord*, Short, Long*, BCD, LBCD*
Relays**	R	**Boolean**
Relay ***	WR	**Word**, DWord*, Short, Long*, BCD, LBCD*
Timer/Counter Contacts	T/C	**Boolean**
Data Registers	DT	**Word**, DWord*, Short, Long*, Float*, BCD, LBCD*
Timer/Counter Registers	SV, EV	**Word**, Short
Index Registers	IX, IY ID	**Word**, Short, **DWord**, Long

*When using these 32 bit data types, two consecutive 16 bit registers will be used; meaning, if address DT1 is declared type DWord, both addresses DT1 and DT2 will be used to store the 32 bit value.
**Bits.
***Words.

Address Specifications

Address Type	Bit Number	Bit Range	Word Number	Word Range
External Inputs*	X<xxx>.<y> xxx-Decimal y-Hex X<xxxxx> xxxxx-Decimal	X000.0-X000.F ... X999.0-X999.F X0-X15999	WX<xxx> xxx-Word Number	WX000-WX999
External Ouputs	Y<xxx>.<y> xxx-Decimal y-Hex Y<xxxxx> xxxxx-Decimal	Y000.0-Y000.F ... Y999.0-Y999.F Y0-Y15999	WY<xxx> xxx-Word Number	WY000-WY999
Internal Relays	R<xxx>.<y> xxx-Decimal y-Hex R<xxxxx> xxxxx-Decimal	R000.0-R000.F ... R999.0-R999.F R0-R15999	WR<xxx> xxx-Word Number	WR000-WR999
Timer/Counter	T<xxx>	T000-T999	N/A	N/A

Address Type	Bit Number	Bit Range	Word Number	Word Range
Contacts*	xxx-Decimal C<xxx> xxx-Decimal	 C000-C999		
Data Registers	N/A	N/A	DT<xxxxx> xxxxx-Decimal	DT00000-DT65535
Special Registers	N/A	N/A	DT<xxxxx> xxxxx-Decimal	DT90000-DT99999
Set Value Area**	N/A	N/A	SV<xxxx> xxxx-Decimal	SV0000-zV9999
Elapsed Value Area**	N/A	N/A	EV<xxxx> xxxx-Decimal	EV0000-EV9999
Index Registers	N/A	N/A	N/A	IX, IY, ID

*Read Only.
**Timers/Counters.

Note: <y> bits are only valid between 0 and F hexadecimal. The bit reference used when accessing X,Y and R memory is only required when using the <xxx>.<y> address format. Normally direct access to X,Y and R can be done using standard Aromat Matsushita/NAIS addressing such as X50, Y122, or R140.

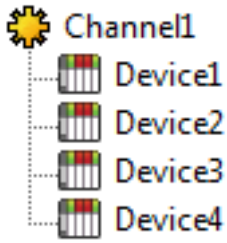
Examples

1. Y15 Output Relay 15.
2. T000 Timer Contact 0.
3. C127 Counter Contact 127.

Optimizing Communications

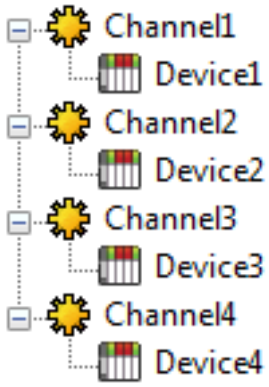
The Aromat Ethernet Driver has been designed to provide the best performance with the least amount of impact on the system's overall performance. While the Aromat Ethernet Driver is fast, there are a couple of guidelines that can be used in order to control and optimize the application and gain maximum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

Our server refers to communications protocols like Aromat 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 Aromat Matsushita/NAIS controller from which data will be collected. While this approach to defining the application will provide a high level of performance, it won't take full advantage of the Aromat 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 Aromat Ethernet channel. 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 Aromat Ethernet Driver could only define one single channel, then the example shown above would be the only option available; however, the Aromat Ethernet Driver can define up to 16 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 16 or fewer devices, it can be optimized exactly how it is shown here.

The performance will improve even if the application has more than 16 devices. While 16 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.

Block Size, which is available on each defined device, can also affect the Aromat Ethernet Driver's performance. Block Size refers to the number of bytes that may be requested from a device at one time. To refine the performance of this driver, configure Block Size to one of the following settings: 32, 64, 128, 256, or 512 bytes. Depending on the Aromat Ethernet device model, the Block Size setting affect the application's performance drastically. A default value of 64 bytes is recommended. If an application has large requests for consecutively ordered data, however, block size should be increased.

Error Descriptions

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

Address Validation

Missing address
Device address '<address>' contains a syntax error
Address '<address>' is out of range for the specified device or register
Data Type '<type>' is not valid for device address '<address>'
Device address '<address>' is Read Only
Array size is out of range for address '<address>'
Array support is not available for the specified address: '<address>'

Device Status Messages

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

Driver Error Messages

Winsock initialization failed (OS Error = n)
Winsock V1.1 or higher must be installed to use the Aromat Ethernet device driver
Unable to bind IP '<IP in hex>' to Port '<Port>' - address is in use by another application
Response to <read/write> request to '<tag address>' on device '<device name>' contained error code '<xx>'
Response to <read/write> request to '<tag address>' on device '<device name>' had an unexpected format
Response to <read/write> request to '<tag address>' on device '<device name>' failed BCC check

Missing address

Error Type:

Warning

Possible Cause:

A tag address that has been specified dynamically has no length.

Solution:

Re-enter the address in the client application.

Device address '<address>' contains a syntax error

Error Type:

Warning

Possible Cause:

A tag address that has been specified dynamically contains one or more invalid characters.

Solution:

Re-enter the address in the client application.

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

Error Type:
Warning

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

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

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

Error Type:
Warning

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

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

Device address '<address>' is Read Only

Error Type:
Warning

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

Solution:
Change the access mode in the client application.

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

Error Type:
Warning

Possible Cause:
A tag address that has been specified dynamically 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 dynamically 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.

'Device <Device name>' is not responding

Error Type:
Serious

Possible Cause:
1. The connection between the device and the Host PC is broken.
2. The IP address assigned to the device is incorrect.
3. The response from the device took longer to receive than the amount of time specified in the "Request Timeout" device setting.

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

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

Error Type:
Serious

Possible Cause:
1. The connection between the device and the Host PC is broken.
2. The named device may have been assigned an incorrect IP address.

Solution:
1. Verify the cabling between the PC and the PLC device.
2. Verify the IP address given to the named device matches that of the actual device.

Winsock initialization failed (OS Error = n)

Error Type:

Fatal

OS Error	Indication	Possible Solution
10091	Indicates that the underlying network subsystem is not ready for network communication.	Wait a few seconds and restart the driver.
10067	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 Aromat Ethernet device driver

Error Type:
Fatal

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.

Unable to bind IP '<IP in hex>' to Port '<Port>' - address is in use by another application

Error Type:
Serious

Possible Cause:
The driver was unable to bind the specified source port and IP because another application is already using this combination of protocol (UDP or TCP/IP), IP and port number.

Solution:
Aromat Ethernet requires the driver to bind the connection's source IP and port number when using UDP or TCP/IP Full Passive. If this is not possible, there are three solutions:

1. Determine what application is using the specified IP and port and then re-configure it to use another address.
2. Use a different combination of source IP and port.
3. Use the TCP/IP Unpassive connection type.

Response to <read/write> request to '<tag address>' on device '<device name>' contained error code '<xx>'

Error Type:
Serious

Possible Cause:
The device returned an error code in response to a read or write request.

Solution:
Consult the Aromat Matsushita/NAIS documentation for meaning of error code and take appropriate corrective measures.

Response to <read/write> request to '<tag address>' on device '<device name>' had an unexpected format

Error Type:
Serious

Possible Cause:
The response to a read or write request was not in the correct format.

Solution:
1. Verify the cabling between the PC and the PLC device.
2. If problem does not appear to be related to noise, contact Technical Support.

Response to <read/write> request to '<tag address>' on device '<device name>' failed BCC check

Error Type:
Serious

Possible Cause:
The received byte check code was incorrect, indicating a corrupted response.

Solution:
Verify the cabling between the PC and the PLC device. Take appropriate shielding measures to reduce electrical noise.

Index

A

B

C

D

E

F

I

L

M

N

O

R

S

T

U

W