The Ethernet/IP plugin
PRINTED MANUAL

© 1999-2024 AGG Software

Ethernet/IP plugin

© 1999-2024 AGG Software

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Printed: 2/2/2024

Publisher

AGG Software

Production

© 1999-2024 AGG Software
http://www.aggsoft.com

Table of Contents

Part 1 Introduction 1

Part 2 System requirements 1

Part 3 Installing Ethernet/IP 1

Part 4 Glossary 2

Part 5 User Manual 3

1 Data query 3
2 Request method 5
3 AB Micro 800 6
4 AB MicroLogix 1400 8
5 AB Logix 5550 10

© 1999-2024 AGG Software

1 Introduction

This parser plugin allows to read tag values from ControlLogix, PLC 5, SLC 500 and MicroLogix controllers using EtherNet/IP protocol. The parser uses "Unconnected Messages" with PCCC-style commands. EtherNet/IP is an industrial network protocol that adapts the Common Industrial Protocol to standard Ethernet.

Features:

- Can send valid data request to any EtherNet/IP-compatible device;
- Symbolic or class/instance/attribute addressing;
- Supports various data types: STRING, BOOL, CONTROL, COUNTER, DINT, INT, LINT, REAL, SINT, TIMER, USINT, UINT, UDINT, ULINT, LREAL, STRING, DATETIME, DATE, TIME;
- Automatically detects a data type of returned data.
- Can read arrays.
- Can poll data by a custom interval.

2 System requirements

The following requirements must be met for "Ethernet/IP" to be installed:

Operating system: Windows 2000 SP4 and above, including both x86 and x64 workstations and servers. The latest service pack for the corresponding OS is required.

Free disk space: Not less than 5 MB of free disk space is recommended.

Special access requirements: You should log on as a user with Administrator rights in order to install this module.

The main application (core) must be installed, for example, Advanced Serial Data Logger.

3 Installing Ethernet/IP

1. Close the main application (for example, Advanced Serial Data Logger) if it is running;
2. Copy the program to your hard drive;
3. Run the module installation file with a double click on the file name in Windows Explorer;
4. Follow the instructions of the installation software. Usually, it is enough just to click the "Next" button several times;
5. Start the main application. The name of the module will appear on the "Modules" tab of the "Settings" window if it is successfully installed.

If the module is compatible with the program, its name and version will be displayed in the module list. You can see examples of installed modules on fig.1-2. Some types of modules require additional configuration. To do it, just select a module from the list and click the "Setup" button next to the list. The configuration of the module is described below.

© 1999-2024 AGG Software

You can see some types of modules on the "Log file" tab. To configure such a module, you should select it from the "File type" list and click the "Advanced" button.

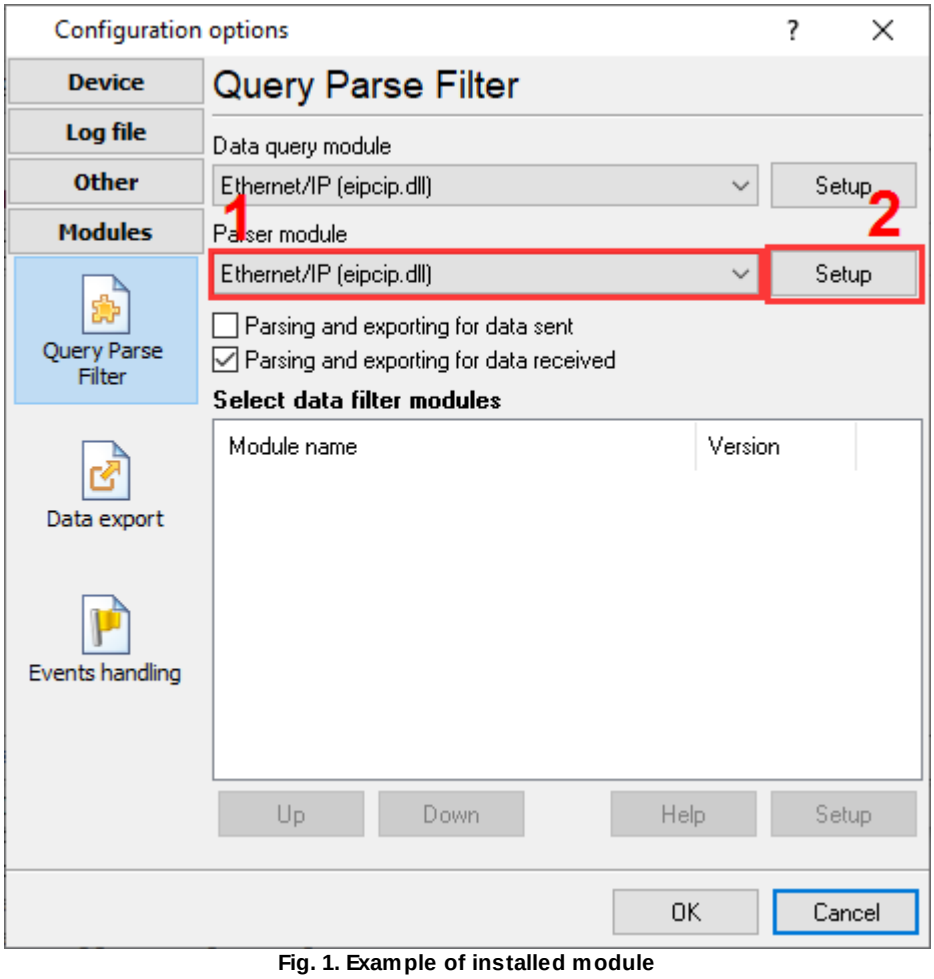

Fig. 1. Example of installed module

4 Glossary

Main program - it is the main executable of the application, for example, Advanced Serial Data Logger and asdlog.exe. It allows you to create several configurations with different settings and use different plugins.

Plugin - it is the additional plugin module for the main program. The plugin module extends the functionality of the main program.

Parser - it is the plugin module that processes the data flow, singling out data packets from it, and then variables from data packets. These variables are used in data export modules after that.

Core - see "Main program."

© 1999-2024 AGG Software

5 User Manual

5.1 Data query

To add new item click "Actions - Add new request". The dialog window will be shown (fig. 2). Enter a request description, that can contain any characters and click the "OK" button.

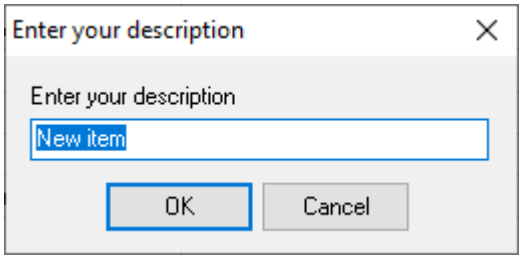

Fig. 1. Name dialog

The new request will appear in the requests tree (fig. 2). Each request has few important options:

Device path - a path to your EIP compatible PLC in the Ethernet/IP network.

It should be empty or 1 for direct connections.

Typically, the path is the **slot number** of the processor module in the backplane (chassis). The empty value means the default slot #0.

But if your communications card is not in the same chassis as your processor, this is the path through the chassis to get to your processor. You will have to add a 1 for every chassis you go through:

Example

Chassis 1: ENBT card in Slot 0
Path would be: { 0 }
Device path parameter: 0

Chassis 1: ENBT card in Slot 1
Path would be: { 1 }
Device path parameter: 1

Chassis 1: ENBT card in Slot 1 (slot is irrelevant), ControlNet Card in Slot 2
Chassis 2: L61 in Slot 4
Path would be: { 2, 1, 4 }
Device path parameter: 2,1,4

Address - it's an address (symbolic name) of a data item in the PLC's memory. If the item is an array then you may add an index in this array.

Examples:

VALUE1 - read data from the "VALUE1" tag.

© 1999-2024 AGG Software

VALUE1[2] - VALUE1 is an array and the program will read the 3rd element from this array (the array index starts from 0).
@22/1/1 - read data from class 22, instance 1 and attribute 1.

User-defined string data types

For example, let's say you have defined a custom string data type with a non-standard data length. In this case, the PLC exports this value as a general structure. If you set the "Data type" parameter to "String", our program will try to interpret the given structured value as a simple string data type.

Bitmask

A bitmask in PLC refers to a binary value used for data manipulation or bitwise operations. It is a technique that involves using specific bits within a number to represent certain conditions or states. By setting or clearing individual bits within a bitmask, you can control various aspects of a PLC program.

For example, if you have a decimal value of 15, it can be interpreted as a bitmask with the value 00001111, where bits 0-3 are set.

To extract an individual bit or bit range, set the "Data type" parameter to "Boolean" and include a logical path in the address field:

VALUE1#0 - extract bit #0 from a numeric value VALUE1.
VALUE1#4 - extract bit #4.
VALUE1#0:4 - extract bits from 0 to 4.

Number of elements to read - for arrays you may read the specified number of sequential elements.

Request timeout - after reaching the timeout limit the program will automatically cancel current request and execute next request in the queue. The timeout value depends on the network on which master (program) and slave (device) is running. If the network is slow then timeout value should be larger and if network is fast then timeout value can be small.

Export name - if this value is not empty the program will export the tag's value using this name. If the name is empty the the program will using the value from the address field as a name.

Scale - if this parameters is not equal 1 then the program will scale a returned numerical value using this coefficient.

Default value - this value will be used if the parser can't get the specified values from a response.

© 1999-2024 AGG Software

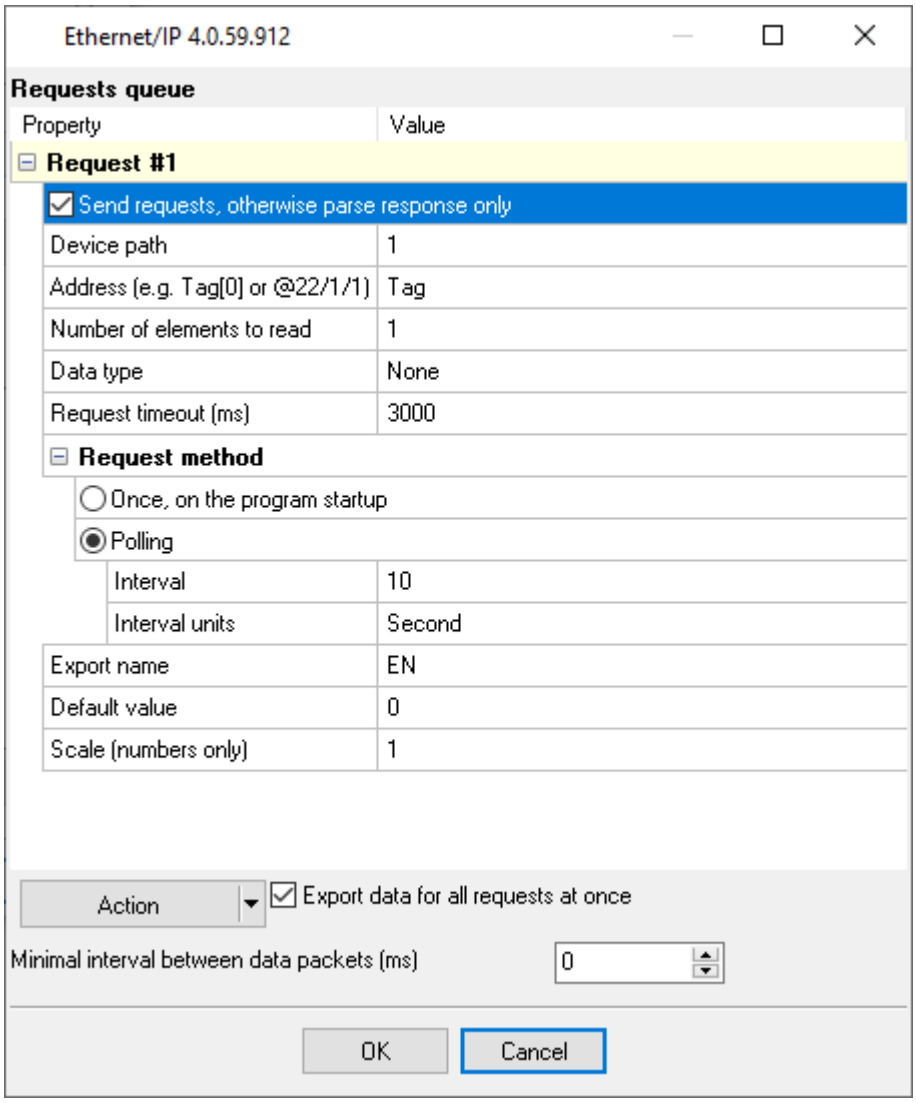

Fig. 2. Requests

5.2 Request method

The plugin can send requests in two modes:

- **Once, on program startup** - the program will send request once, when the program starts.
- **Polling** - the program will be sending request periodically based on an interval specified. The interval between requests depends on the network on which master (program) and slave (device) is running. If the network is slow then time for each request will be larger and vice versa. Because, the program are executing all requests in the queue one by one, then time between requests depends on the number of requests in the queue.

© 1999-2024 AGG Software

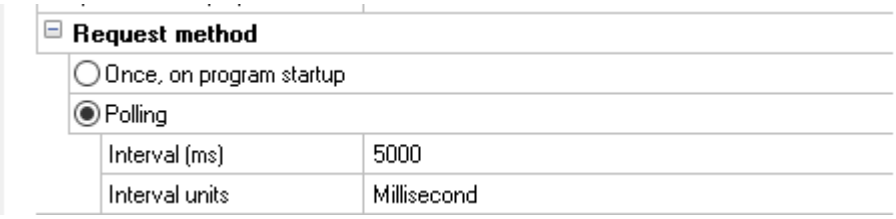

Fig.3. Request methods

If you added few requests to the queue, then you can move it up and down. Select a request title and execute a corresponding menu item by clicking the "Actions" button.
With help of this button you can change an item description and delete requests.

You can access all actions through the popup menu in the request tree.

5.3 AB Micro 800

This PLC series uses a symbolic addressing method. The program may read data from the "Global variables" area by a tag name. The logger should work in the TCP client mode and connect to a port #44818, directly to the PLC.

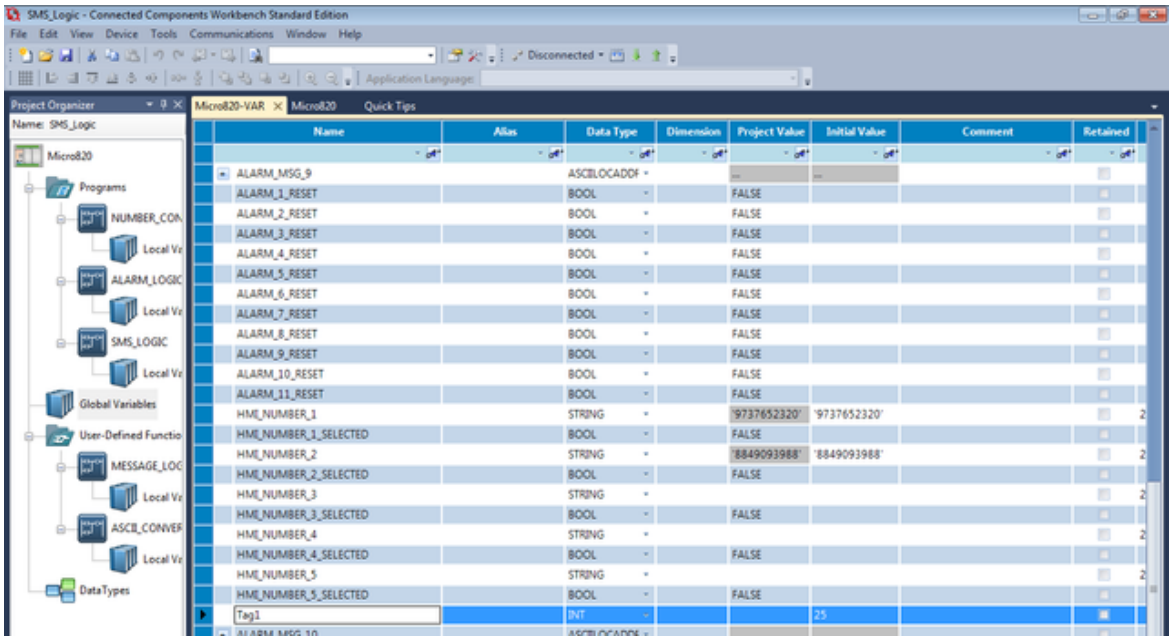

Fig. 3. Tag in a PLC

© 1999-2024 AGG Software

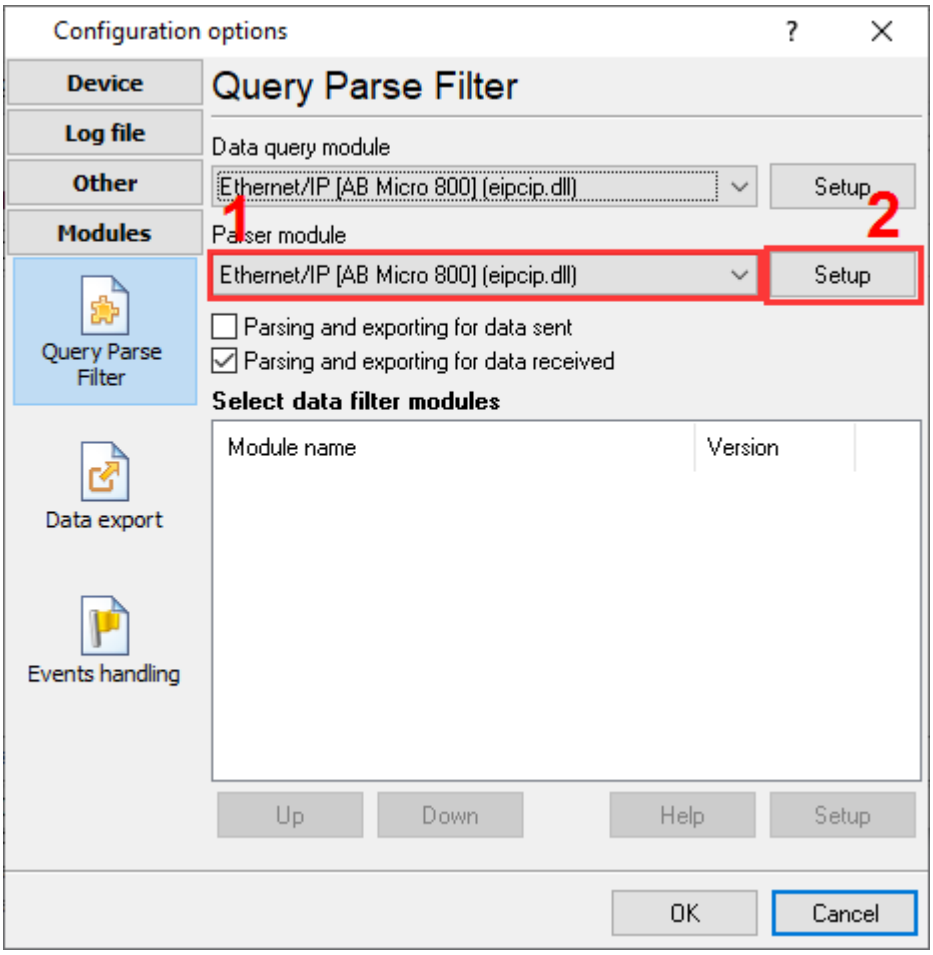

Fig. 4. Data parser

© 1999-2024 AGG Software

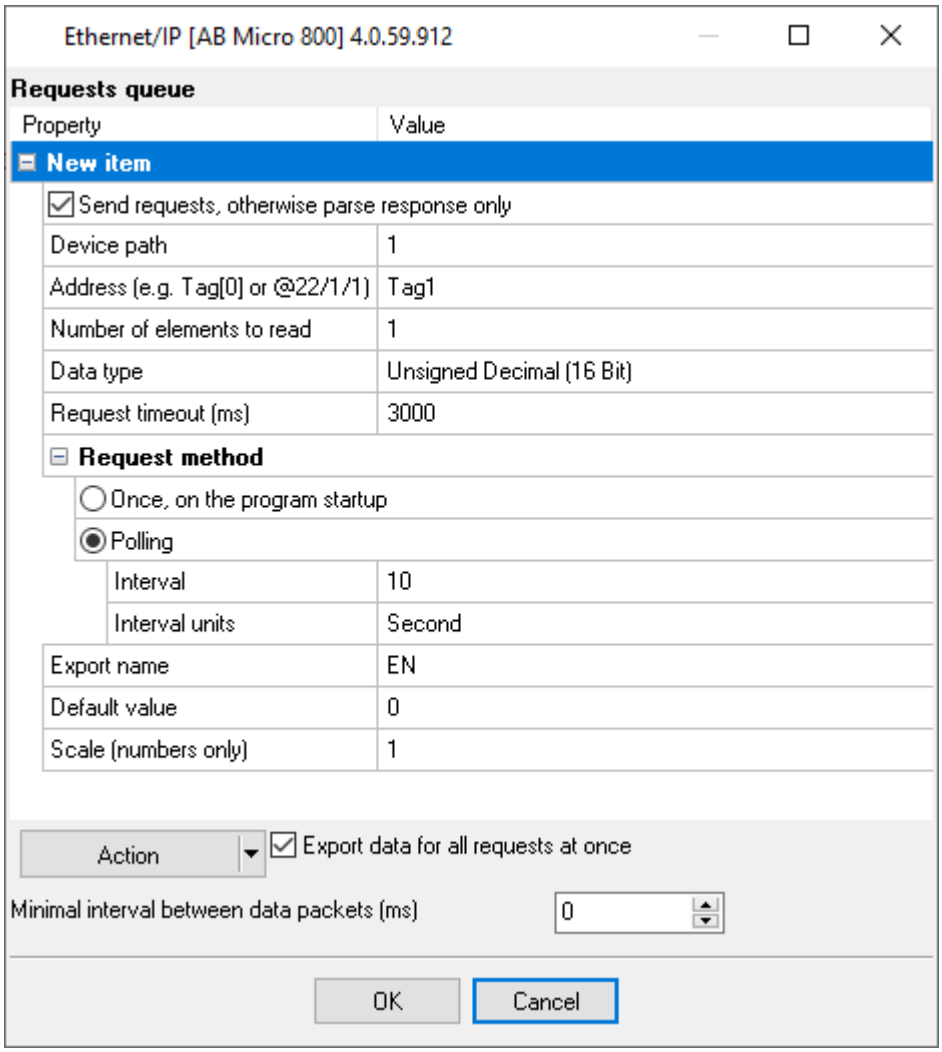

Fig. 5. Queue

5.4 AB MicroLogix 1400

This PLC series does not support a symbolic addressing method. The program may read data from area by a file type and address. The following file types are supported:

R - Control;
C - Counter;
F - Floating-point;
I - Input;
N - Integer;
O - Output;
T - Timer;

The data address should look like: N7:0

N - file data type ID.
7 - file number.

© 1999-2024 AGG Software

0 - element address

The logger should work in the TCP client mode and connect to a port #44818, directly to the PLC.

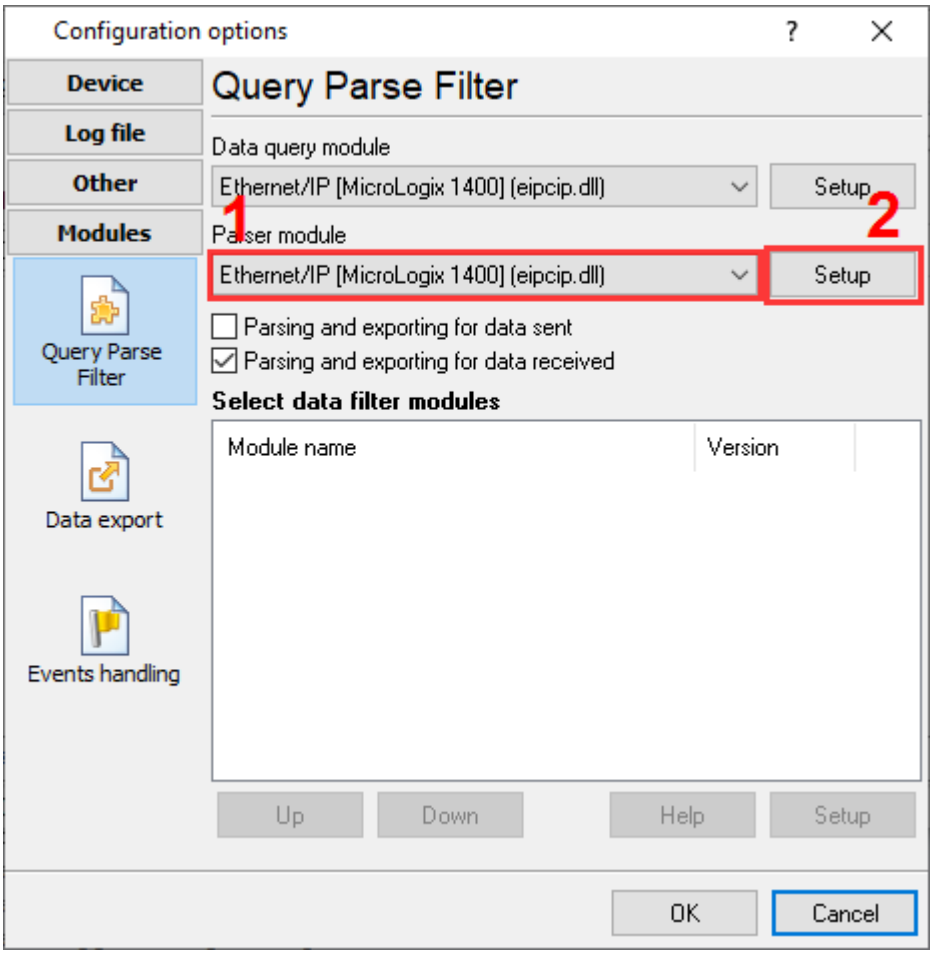

Fig. 6. Data parser

© 1999-2024 AGG Software

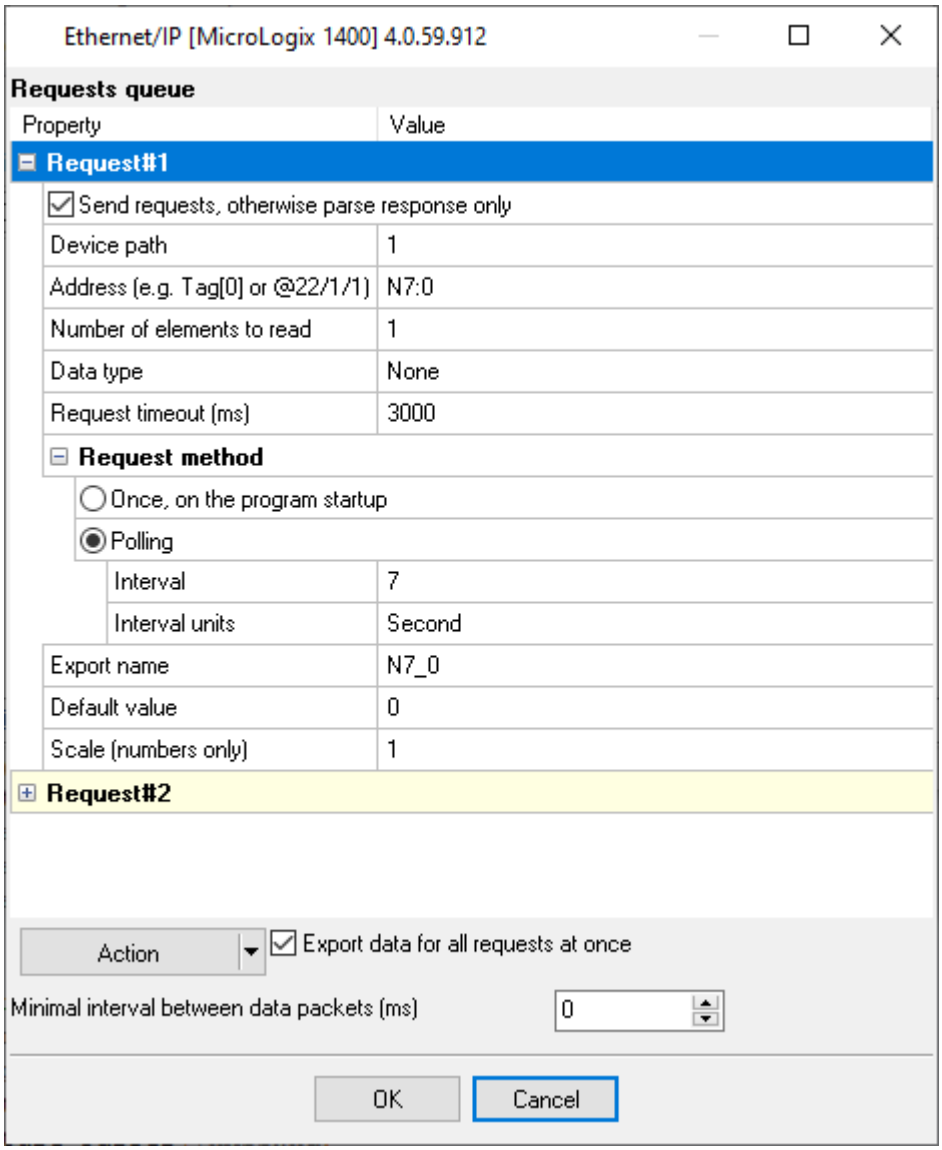

Fig. 7. Queue

5.5 AB Logix 5550

This PLC series supports a symbolic (logical) addressing method. When the program connects to this PLC first time, it reads a list of all available PLC's variables with its names, memory location and data types. After this, internally, the plugin will convert a tag name to an instance ID for faster data reading. Furthermore, it increases reading rate because an instance ID allocates fewer bytes in a request, plus, you can combine several tags in one request.

The logger should work in the TCP client mode and connect to a port #44818.

Note: In this example, the program reads data from Slot 1 at Chassis 1.

© 1999-2024 AGG Software

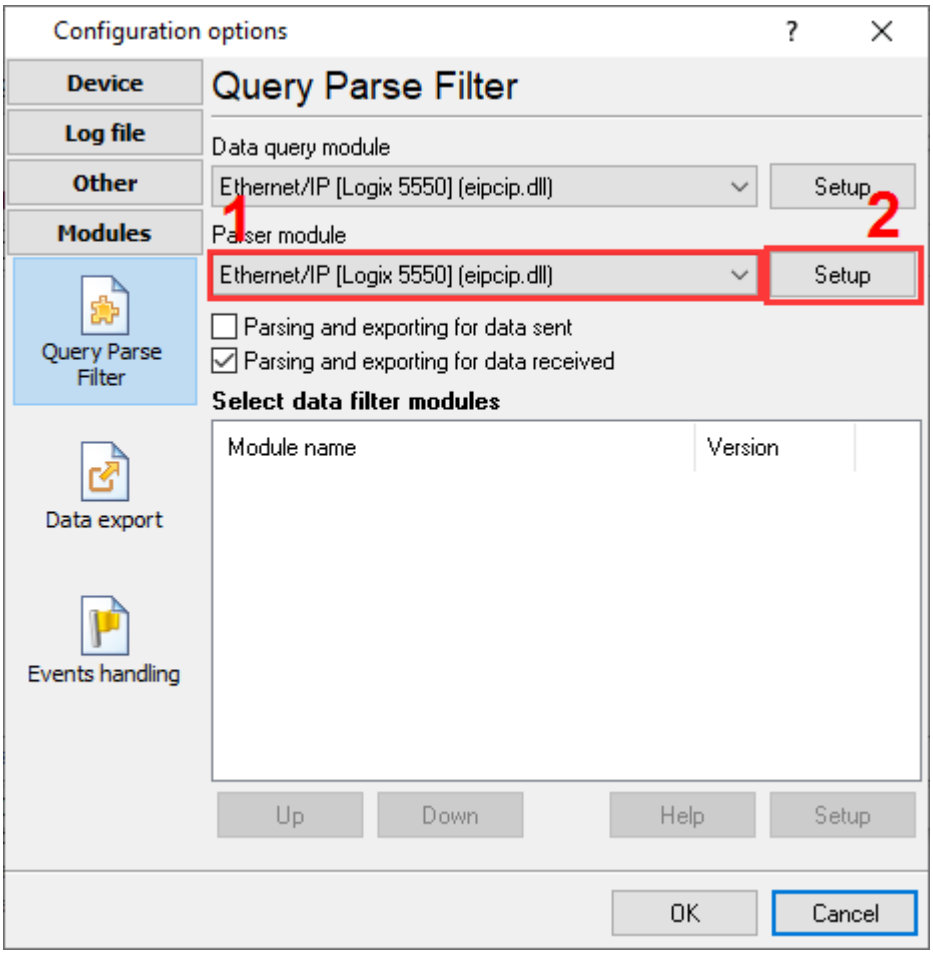

Fig. 8. Parser selection

Example 1: read one tag in a request

This method is useful if you need to process each tag individually, scale a value or export with a unique name.

© 1999-2024 AGG Software

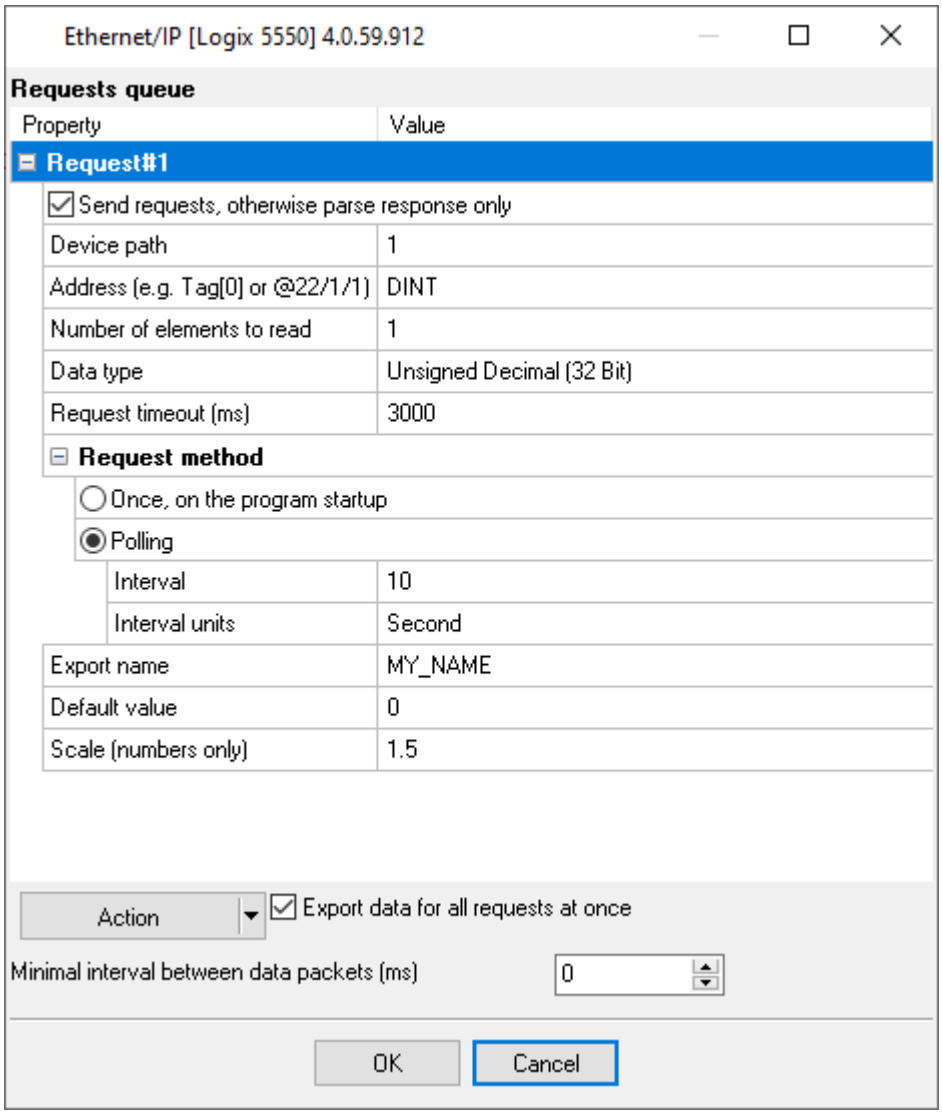

Fig. 9. Queue

Parsed data example:

DATE_TIME_STAMP[7]=2024-09-26 14:20:07;DATA_SOURCE_FULL_NAME[8]="10.107.89.30:44818";DATA_SOURCE_NAME[8]="172.0.0.1:44818";EIP_DEVICE[8]="1756-ENBT/A";EIP_DEVICE_SN[3]=1111111;EIP_REQ[8]="Request1";EIP_ADDRESS[8]="DINT";DINT[3]=19582

Example 2: combine tags with identical data types

Grouping data in one request increases reading rate. But in this case, you cannot specify an individual scale factor for all tags.

© 1999-2024 AGG Software

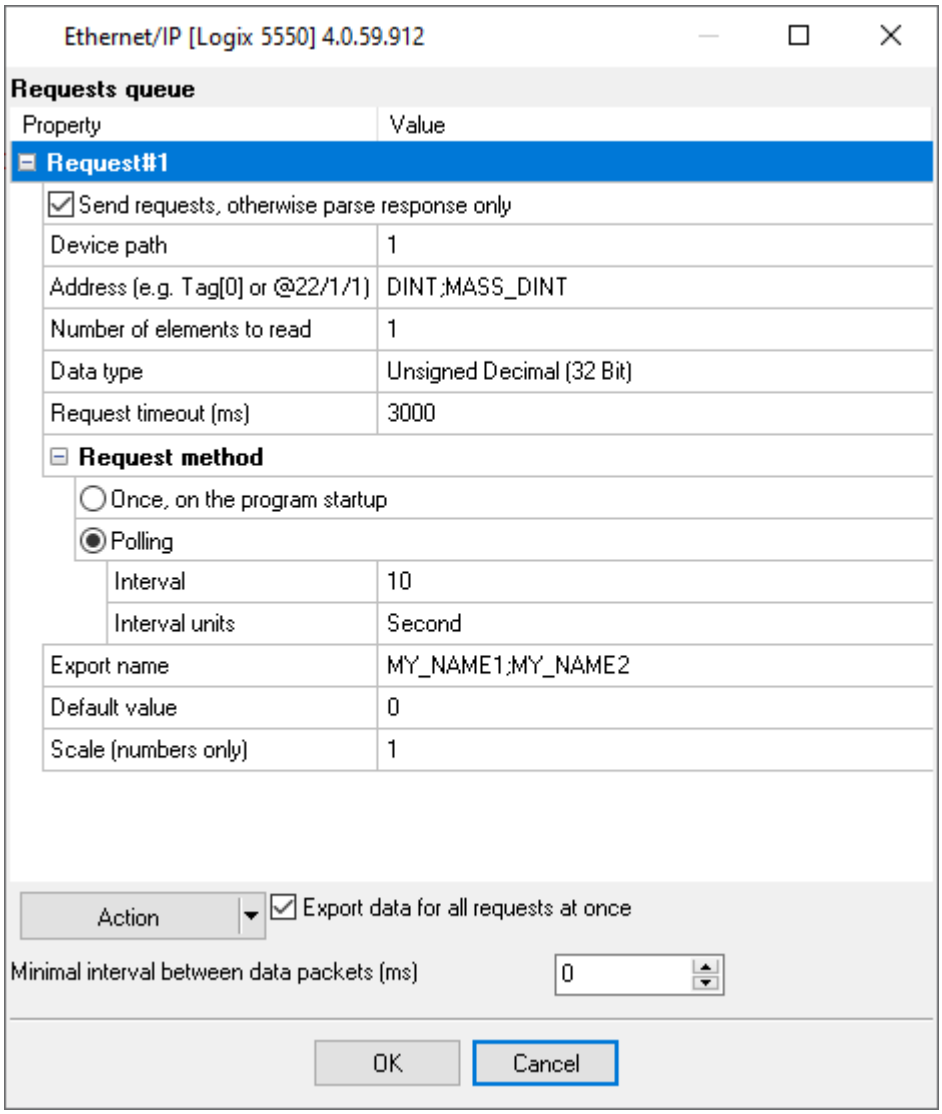

Fig. 10. Queue

Parsed data example:

DATE_TIME_STAMP[7]=2024-09-26 14:20:07;DATA_SOURCE_FULL_NAME[8] ="10.107.89.30:44818";DATA_SOURCE_NAME[8]="172.0.0.1:44818";EIP_DEVICE[8]="1756-ENBT/A";EIP_DEVICE_SN[3]=1111111;EIP_REQ[8]="Request1";EIP_ADDRESS[8] ="DINT;MASS_DINT";DINT[3]=19582;MASS_DINT[3]=123

Example 3: combine different tags in one request

This method works for simple data types only (e.g. INT, DINT, REAL, STRING, etc.). You cannot combine simple data type and structured (user-defined) data types in one request.

© 1999-2024 AGG Software

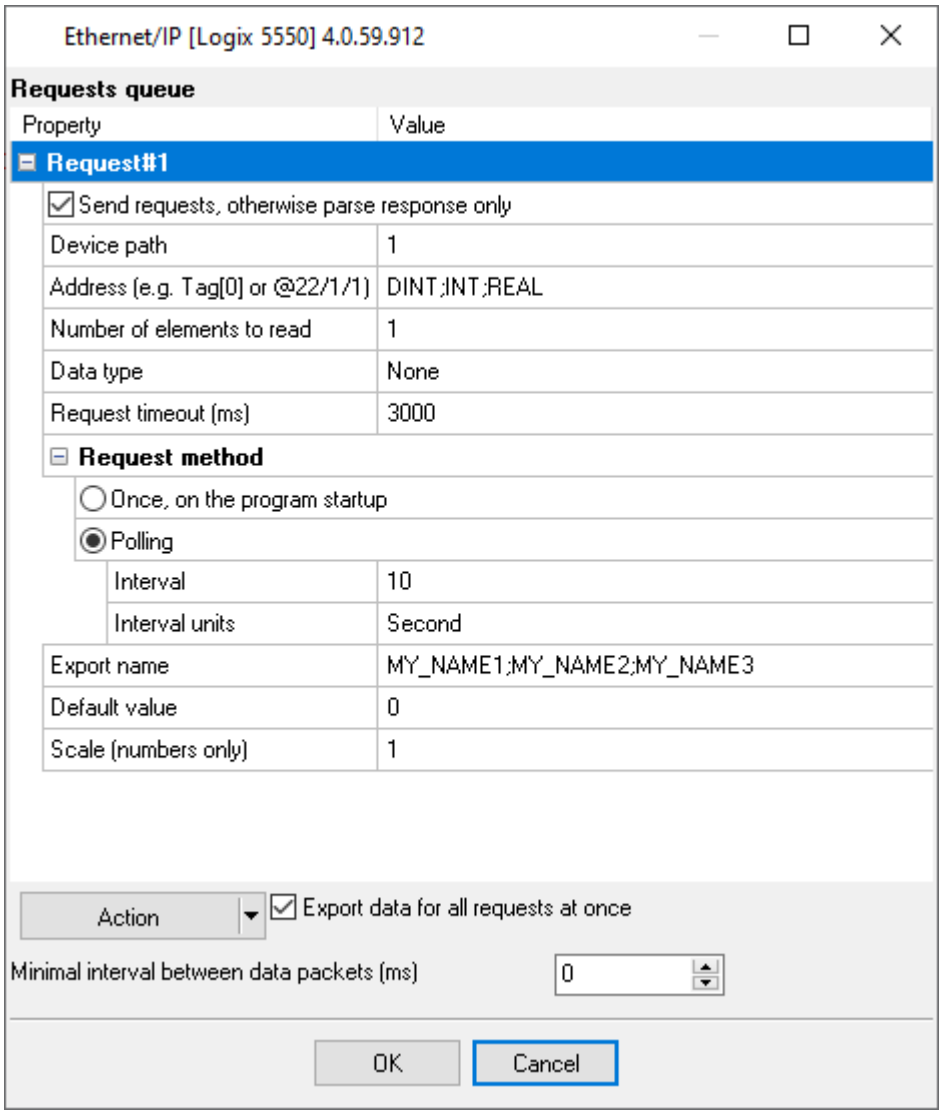

Fig. 11. Queue

Parsed data example:

DATE_TIME_STAMP[7]=2024-09-26 14:20:07;DATA_SOURCE_FULL_NAME[8]
="10.107.89.30:44818";DATA_SOURCE_NAME[8]="172.0.0.1:44818";EIP_DEVICE[8]="1756-
ENBT/A";EIP_DEVICE_SN[3]=1111111;EIP_REQ[8]="Request1";EIP_ADDRESS[8]
="DINT;MASS_DINT;REAL";VALUE1[3]=19582;VALUE2[3]=123;VALUE3[5]=789.455993652344

Example 4: reading an attribute of a value with structured data type

This method is used to read a value of complex (structured) data types.

© 1999-2024 AGG Software

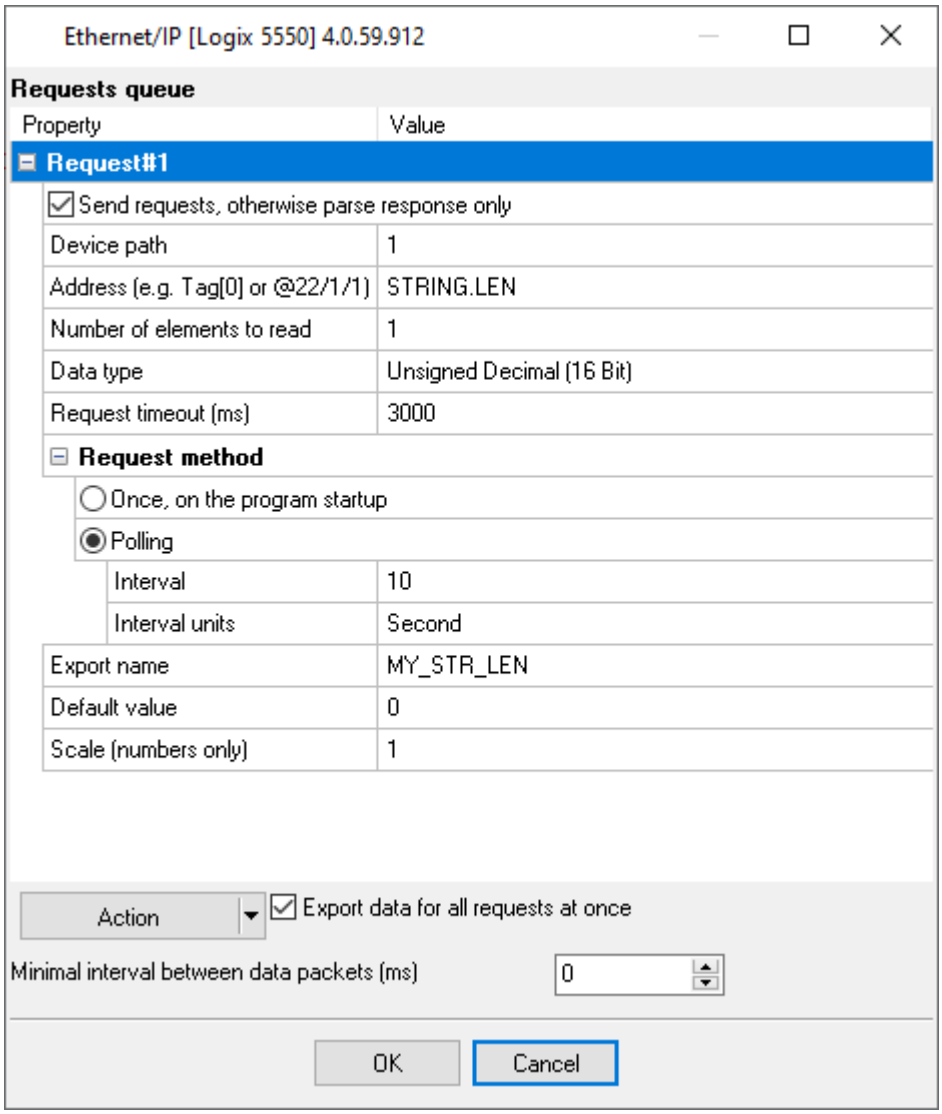

Fig. 12. Queue

© 1999-2024 AGG Software