# 460TCPWI-NNA1 Protocol Gateway

## Product User Guide

*Firmware Version 8.7.22*

**Trademarks**

CompactLogix, ControlLogix, & PLC-5 are registered trademarks of Rockwell Automation, Inc. EtherNet/IP is a trademark of the ODVA. MicroLogix, RSLogix 500, and SLC are trademarks of Rockwell Automation, Inc. Microsoft, Windows, and Internet Explorer are registered trademarks of Microsoft Corporation. BACnet® is a registered trademark of American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE). All other trademarks and registered trademarks are the property of their holders.

**Limited Warranty**

Real Time Automation, Inc. warrants that this product is free from defects and functions properly.

EXCEPT AS SPECIFICALLY SET FORTH ABOVE, REAL TIME AUTOMATION, INC. DISCLAIMS ALL OTHER WARRANTIES, BOTH EXPRESSED AND IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR APPLICATION. THIS LIMITED WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS. YOU MAY ALSO HAVE OTHER RIGHTS, WHICH VARY FROM STATE TO STATE.

The examples and diagrams in this manual are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular application, Real Time Automation, Inc. cannot assume responsibility or liability for actual use based on the examples and diagrams. Except as specifically set forth above, Real Time Automation and its distributors and dealers will in no event be liable for any damages whatsoever, either direct or indirect, including but not limited to loss of business profits, income, or use of data. Some states do not allow exclusion or limitation of incidental or consequential damages; therefore, the limitations set forth in this agreement may not apply to you.

No patent liability is assumed by Real Time Automation with respect to use of information, circuits, equipment, or software described in this manual.

Government End-Users

If this software is acquired by or on behalf of a unit or agency of the United States Government, this provision applies: The software (a) was developed at private expense, is existing computer software, and was not developed with government funds; (b) is a trade secret of Real Time Automation, Inc. for all purposes of the Freedom of Information Act; (c) is “restricted computer software” submitted with restricted rights in accordance with subparagraphs (a) through (d) of the Commercial “Computer Software-Restricted Rights” clause at 52.227-19 and its successors; (d) in all respects is proprietary data belonging solely to Real Time Automation, Inc.; (e) is unpublished and all rights are reserved under copyright laws of the United States. For units of the Department of Defense (DoD), this software is licensed only with “Restricted Rights”: as that term is defined in the DoD Supplement of the Federal Acquisition Regulation 52.227-7013 (c) (1) (ii), rights in Technical Data and Computer Software and its successors, and: Use, duplication, or disclosures is subject to restrictions as set forth in subdivision (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at 52.227-7013. If this software was acquired under GSA schedule, the U.S. Government has agreed to refrain from changing or removing any insignia or lettering from the Software or documentation that is provided or from producing copies of the manual or media. Real Time Automation, Inc.

© 2021 Real Time Automation, Inc. All rights reserved.

Revision History ..... 6
Overview ..... 7
Hardware Platforms ..... 8
Hardware – NNA1 ..... 9
Powering the Gateway ..... 9
Mounting with a DIN Rail ..... 10
Installing ..... 10
Removing ..... 10
Accessing the Main Page ..... 11
Error: Main Page Does Not Launch ..... 12
Committing Changes to the Settings ..... 13
Main Page ..... 14
Device Configuration ..... 15
Network Configuration ..... 16
TCP/IP Configuration ..... 17
ASCII Configuration ..... 18
Receive Data ..... 19
Receive Data – Operation Mode ..... 20
Transmit Data ..... 21
Transmit Data – Triggering Methods ..... 22
ASCII Configuration – Technology Triggering Method ..... 24
ASCII Configuration – ASCII Parsing ..... 26
ASCII Configuration – ASCII Parsing Examples ..... 27
ASCII Configuration – ASCII Concatenating ..... 29
ASCII Configuration – ASCII Concatenating Examples ..... 31
ASCII Configuration – ASCII Message Counter ..... 33
Web Interface Configuration ..... 34
Web Interface Gateway Server Configuration ..... 34
Web Interface Gateway Client Configuration ..... 34
Automatic Data Transfer to User Host (Used as a Web Client) ..... 35
Operation Mode ..... 36
Web Interface Data Point List Configuration ..... 37
Web Interface Data Point List Configuration: Auto-Configure (Default) ..... 38
Auto-Configure Group by Device vs. Auto-Configure Group by Data Type ..... 39

Group by Device (Default Method) ..... 39
Group by Data Type ..... 39
Web Interface Data Point List Configuration: Manual Mode ..... 40
Configuring Read and Write Data Points ..... 40
Web Interface Group Configuration ..... 42
Requirements for Device and Group Names: ..... 44
Web Interface: XML/JSON Data Format ..... 46
Retrieving Data from the Gateway from a Web Client ..... 46
Basics ..... 46
Requesting Data for an Individual Device ..... 47
Requesting a Group of Devices ..... 47
Filtering Requests by Point Names ..... 47
Advanced (Optional XML Data Direction Annotation) ..... 49
Description of Format – Both XML and JSON ..... 50
XML Output ..... 51
JSON Output ..... 51
Writing Data to the Gateway ..... 52
Which method to use? ..... 52
The multipart/form data Method ..... 52
The application/x-www-form-urlencoded Method ..... 54
Minimum XML Input ..... 55
Expanded XML Input ..... 55
Special Note for XML Strings ..... 56
Minimum JSON Input ..... 56
Expanded JSON Input ..... 57
Special Note for JSON Strings ..... 57
Mapping - Transferring Data Between Devices ..... 58
Display Mapping and Values ..... 59
Display Data ..... 59
Display String ..... 62
Display String use case ..... 64
Data and String Mapping – Auto-Configure ..... 65
Data Mapping – Explanation ..... 66
Data Mapping – Adding Diagnostic Information ..... 67

String Mapping – Explanation ..... 71
Mapping – Auto-Configure Mode to Manual Configure Mode ..... 72
Mapping – Manual Configure Mode to Auto-Configure Mode ..... 73
View as Text ..... 74
Data Mapping ..... 74
String Mapping ..... 74
Base Triggering – Data Validiation Triggering ..... 75
Security Configuration ..... 77
Security Configuration-Security Levels ..... 78
Security - Log In ..... 79
Security - Log Out ..... 79
Email Configuration ..... 80
Alarm Configuration ..... 81
Diagnostics – Alarm Status ..... 83
Alarms – Active ..... 83
Alarms – Clear ..... 84
Change of State (COS) Configuration ..... 85
Diagnostics Info ..... 86
Diagnostics Mapping ..... 86
Diagnostics – ASCII ..... 87
Diagnostics – Web Interface ..... 92
LED Configuration ..... 95
Configuration Files ..... 96
Export Configuration ..... 96
Import Configuration ..... 96
Save and Replace Configuration Using SD Card ..... 98
Saving Configuration Using SD Card ..... 98
Replacing Configuration Using SD Card ..... 98
Intelligent Reset Button ..... 99
Utilities ..... 100

## Revision History

| Version | Date | Notes |
|---|---|---|
| **8.4.5** | 11/18/2019 | Features Added<br>1. Released OPC UA Server (US) Protocol<br>2. Ability to now Import/Export Template Files with out an FTP session.<br>Bug Fixes<br>1. Updated Profinet Server (PS) on N34 hardware Platform<br>2. Updated Wi-Fi software |
| **8.6.0** | 2/28/20 | Bug Fixes<br>1. Omron Plc Communication fixes for EtherNet/IP<br>2. Profinet GSDML Substitute values fix |
| **8.7.4** | 9/1/20 | Features Added:<br><br>1. BMS, BM, DFM, DS, DM, TCP, USB, PBS have been ported to the latest base software.<br>2. TCP,BMS,BM now Available on N2E and N2EW hardware Platform<br>3. New ASCII Mode Available on TCP/A/USB/WI protocols<br>4. User Guides updated with more examples<br><br>Bug Fixes:<br><br>1. Improved Data Mapping and String Mapping performance<br><br>2. Improved functionality/performance on EC,ETC,ES,MC,MS,BS,BC, A,,WI,PS protocols. |
| **8.7.22** | 4/6/21 | Features Added:<br><br>1. Support for RSLogix Versions 32 + with unsigned data type support<br>2. ETC now support Long integer files (L files) for MicroLogix PLCS that support them<br>3. SC now supports data block (DB) access |

## Overview

The 460TCPWI-NNA1 gateway connects up to 10 Ethernet TCP/IP devices with a Web Interface. By following this guide, you will be able to configure the 460TCPWI-NNA1 gateway.

For further customization and advanced use, please reference the appendices located on the CD or online at: http://www.rtautomation.com/product/460-gateway-support/.

If at any time you need further assistance, do not hesitate to call Real Time Automation support. Support Hours are Monday-Friday 8am-5pm CST

Toll free: 1-800-249-1612
Email: support@rtautomation.com

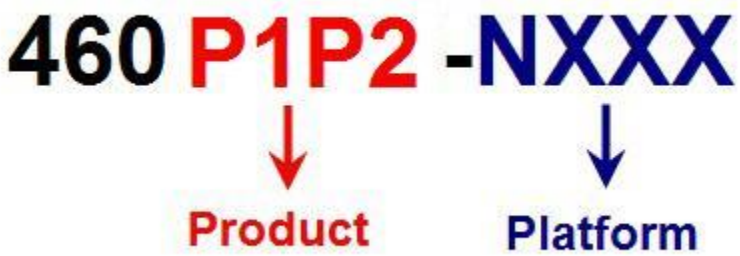
## Hardware Platforms

The 460 Product Line supports a number of different hardware platforms. There are differences in how they are powered, what serial settings are supported, and some diagnostic features supported (such as LEDs). For these sections, be sure to identify the hardware platform you are using.

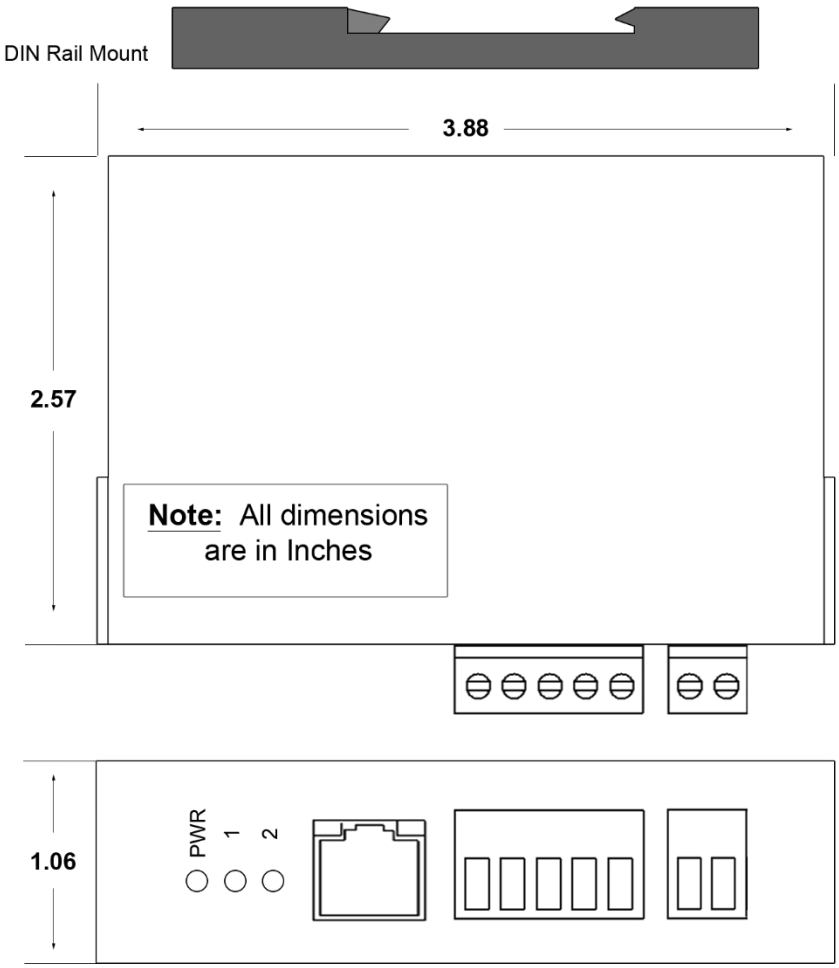
To find which hardware platform you are using:

1) Look on the front or back label of the unit for the part number.
2) On the webpage inside the gateway, navigate to the dropdown menu under **Other** and select **Utilities**. Click the **Listing of Revisions** button. The full part number is displayed here.

Once you have the full part number, the platform will be the number following the "-N":

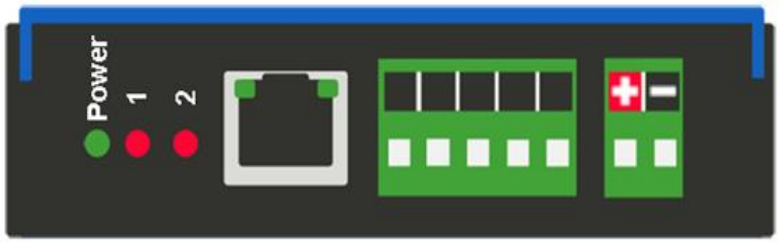


## Hardware – NNA1




## Powering the Gateway

1) Connect a 12-24 VDC power source to the gateway, Red Wire = (+) Black Wire = (-).
   a) The unit draws 175mA @ 12 V.

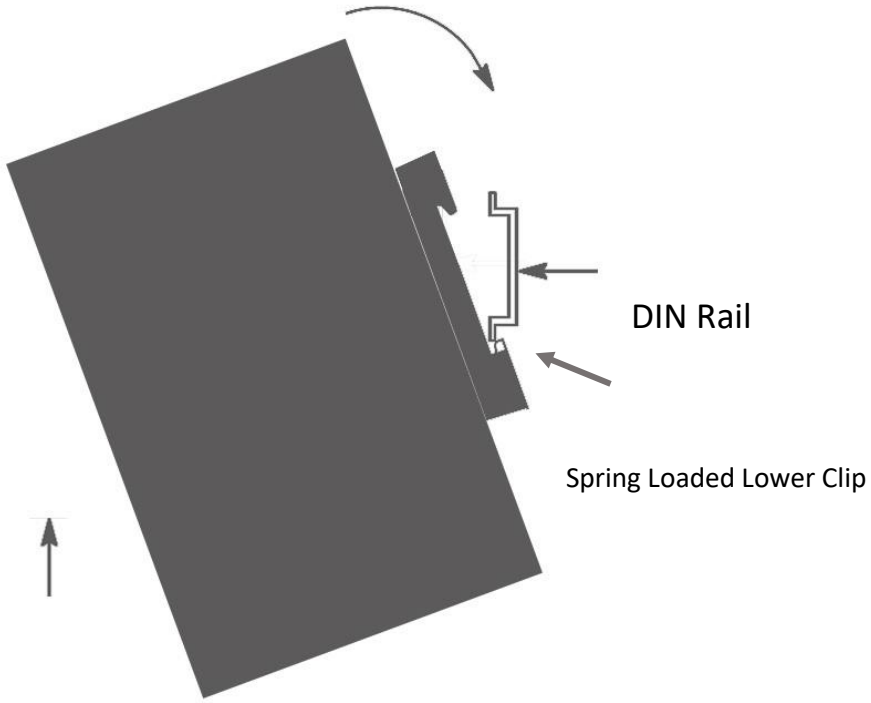


## Mounting with a DIN Rail

### Installing

Follow these steps to install your interface converter.

1) Mount your DIN Rail.
2) Hook the bottom mounting flange under the DIN Rail.
3) While pressing the 460TCPWI-NNA1 against the rail, press up to engage the spring loaded lower clip and rotate the unit parallel to the DIN Rail.
4) Release upward pressure.




### Removing

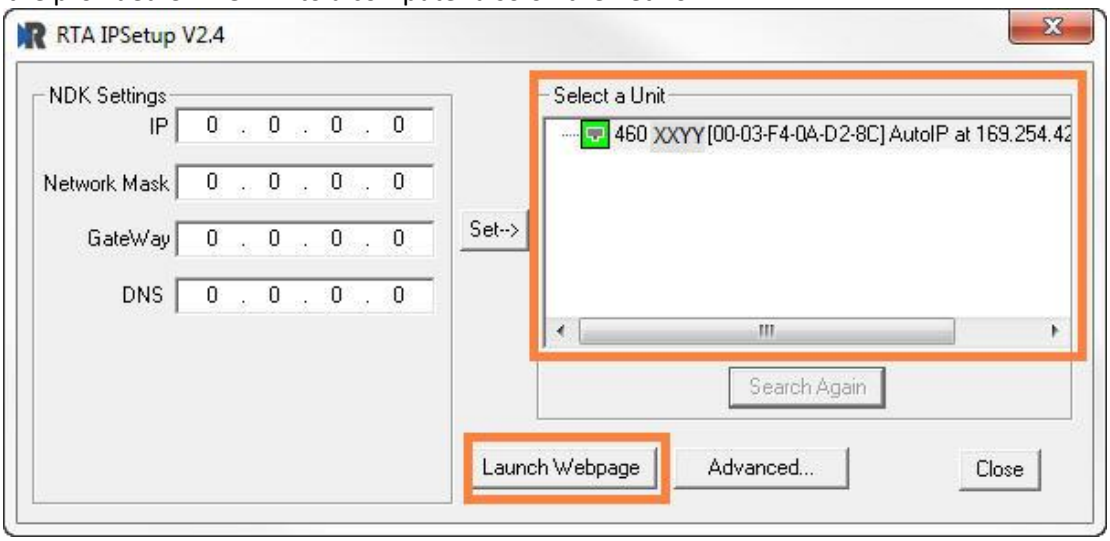
Follow these steps to remove your interface converter.

1) Press up on unit to engage the spring loaded lower clip.
2) Swing top of the unit away from DIN Rail.

## Accessing the Main Page

The following steps will help you access the browser based configuration of the gateway. By default, DHCP is enabled. If the gateway fails to obtain an IP address over DHCP it will Auto IP with 169.254.X.Y. For more information on your Operating system network setting refer to the Access Browser Configuration Doc on the CD or download from our support web site.

1) Insert the provided CD-ROM into a computer also on the network.




2) Run the IPSetup.exe program from the CD-ROM.
3) Find unit under “Select a Unit”.

a. Change Gateway’s IP address to match that of your PC if DHCP has failed.
   i. You will know DHCP has failed if the gateway’s IP address is AutoIP at 169.254.X.Y.
   ii. If successful, it will say DHCP’d at ex: 192.168.0.100 or however your DCHP Client is set up.
b. If you do not see the gateway in this tool, then your PC is most likely set up as a static IP.
   i. Change your PC’s network settings to be DHCP. If DHCP fails, then it will change to be on the 169.254.x.y network.
   ii. Relaunch the IP Setup tool to see if gateway can be discovered now.

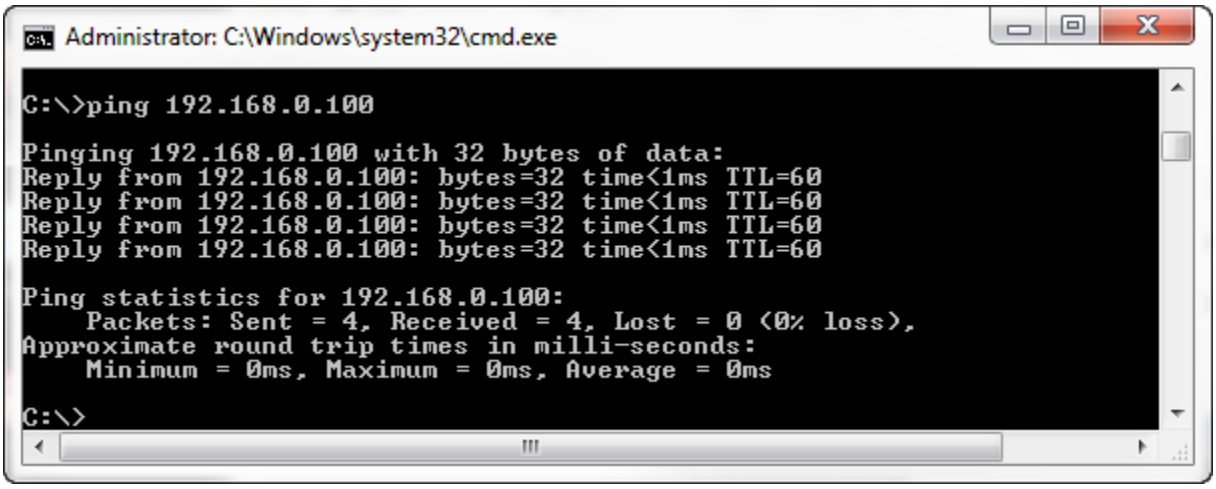
4) Click **Launch Webpage**. The Main page should appear.

**Default setting is set to DHCP. If DHCP fails, default IP Address is 169.254.x.y**

## Error: Main Page Does Not Launch

If the Main Page does not launch, please verify the following:

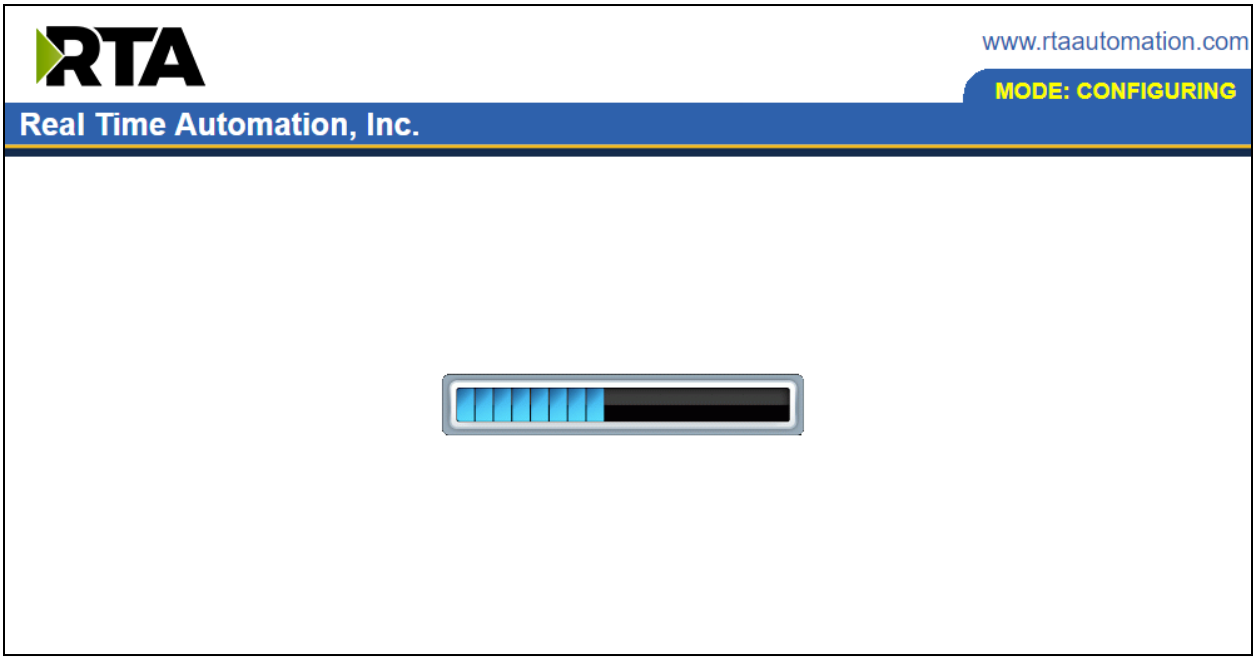
1) Check that the PC is set for a valid IP Address
   a. Open a MS-DOS Command Prompt
   b. Type “ipconfig” and press enter
   c. Note the PC’s IP Address, Subnet, and Default Gateway
2) The gateway must be on the same Network/Subnet as the PC whether it’s setup for DHCP or Static. Once you have both devices on the same network, you should be able to ping the gateway using a MS-DOS Command Prompt.




The Screenshot above shows a gateway that is currently set to a static IP Address of 192.168.0.100. If you are able to successfully ping your gateway, open a browser and try to view the main page of the gateway by entering the IP Address of the gateway as the URL.

## Committing Changes to the Settings

- All changes made to the settings of the gateway in Configuration Mode will not take effect until the gateway is restarted via the webpage. Changes will not be stored if the gateway's power is removed prior to a reboot.
- **NOTE:** The gateway does not need to be restarted after every change. Multiple changes can be made before a restart, but they will not be committed until the gateway is restarted.
- When all desired changes have been made, press the **Restart Now** button.
- The webpage will redirect to our rebooting page shown below:




- The reboot can take up to 20 seconds.
  - If the IP address has not been modified, the gateway will automatically redirect to the main page.
  - If the IP address was modified, a message will appear at the top of the page to instruct the user to manually open a new webpage at that new IP.

## Main Page

The main page is where important information about your gateway and its connections are displayed.

Mode (orange box below):

Running Mode:

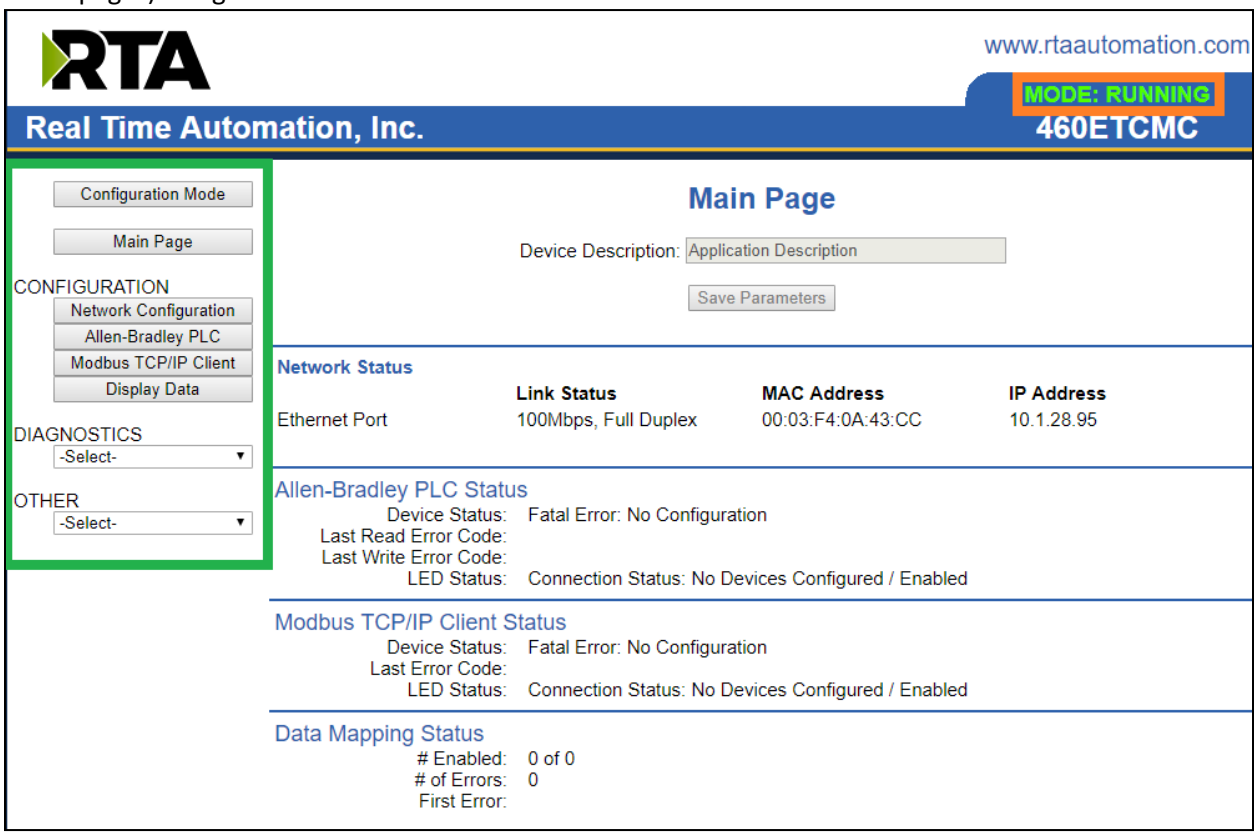
- Protocol communications are enabled
- Configuration cannot be changed during Running Mode. If changes are needed, click the **Configuration Mode** button shown in the green box below

Configuring Mode:

- Protocol communication is stopped and no data is transmitted
- Configuration is allowed

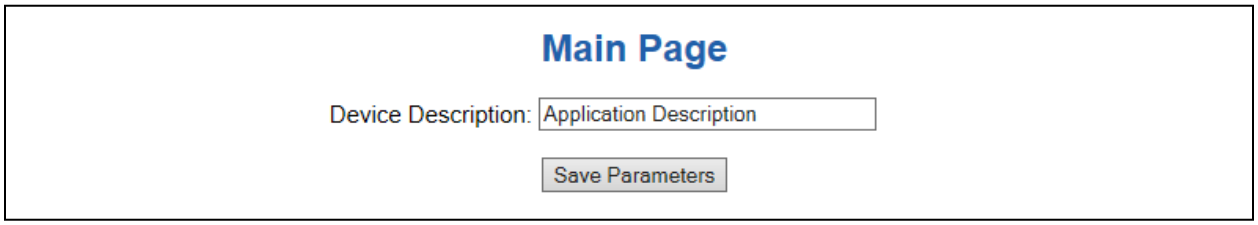
Navigation (green box below):

You can easily switch between modes and navigate between pages (Configuration, Diagnostics, and Other pages) using the buttons on the left hand side.



## Device Configuration

The device configuration area is where you assign the device description parameter. Changes can only be made when the gateway is in Configuration Mode.



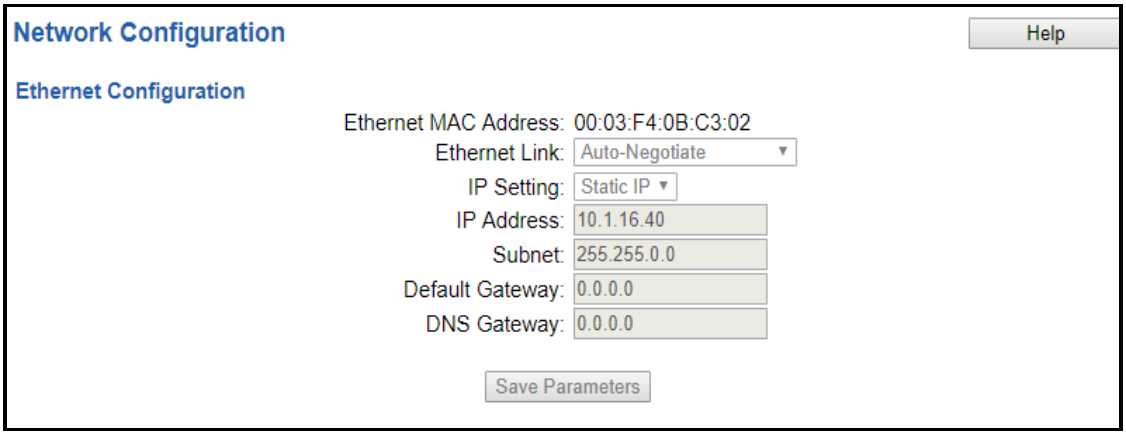

Once you are done configuring the Description, click the **Save Parameters** button.

## Network Configuration

The network configuration area is where you assign the IP address and other network parameters. Changes can only be made when the gateway is in Configuration Mode.

Once you are done configuring the Network Settings, click the **Save Parameters** button.

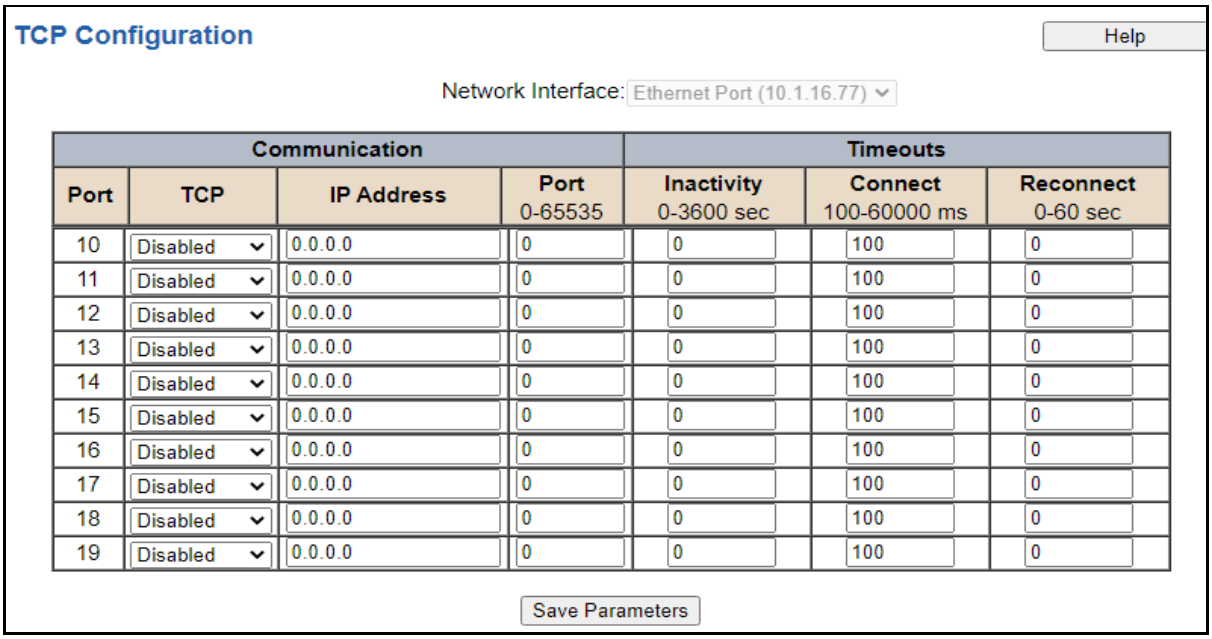
If you are changing the IP Address of the gateway, the change will not take effect until the unit has been rebooted. After reboot, you must enter the new IP Address into the URL.




**It is recommended to leave the DNS Gateway set to 0.0.0.0 and the Ethernet Link as Auto-Negotiate. If configuring the gateway to use E-mail, the DNS Gateway must be set.**

## TCP/IP Configuration

Click the **TCP/IP Configuration** button to access the primary configuration page for this device. If you have an N2E/N2EW hardware, then select which Network Interface ASCII comms will be on.




**To connect to a TCP server device on the network, enable a TCP client connection in the gateway for one of the available ports:**

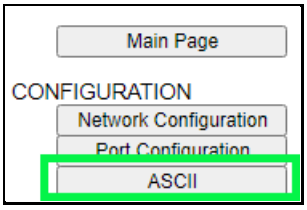
1) **TCP**: Under this dropdown select TCP Client.
2) **IP Address**: Enter the IP address of the TCP server to connect to.
3) **Port**: Enter the port to open a connection.
4) **Inactivity Timeout**: Enter the amount of time, in seconds, the gateway will wait for data to be transmitted to the gateway or sent from the gateway to the TCP device. If the timer expires, the gateway will close the connection.
5) **Connect Timeout**: Enter the amount of time, in milliseconds, the gateway will wait for the TCP server to accept the connection.
6) **Reconnect Timeout**: Enter the amount of time, in seconds, the gateway will wait after the Connect Timeout has expired before issuing another connect.

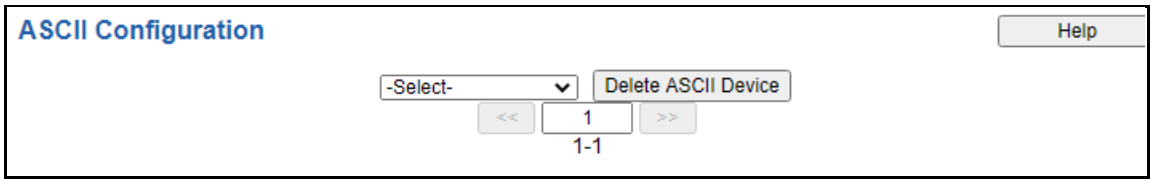
**To connect to a TCP client device on the network, enable a TCP server connection in the gateway for one of the available ports:**

1) **TCP**: Under this dropdown select TCP server.
2) **IP Address**: Enter the IP address of the TCP client that will connect to the gateway.
3) **Port**: Enter the port to listen for a connection.
4) **Inactivity Timeout**: Enter the amount of time, in seconds, the gateway will wait for data to be transmitted to the gateway or sent from the gateway to the TCP device. If the timer expires, the gateway will close the connection.
5) **Connect Timeout**: Not used in this case (Client only).
6) **Reconnect Timeout**: Not used in this case (Client only).

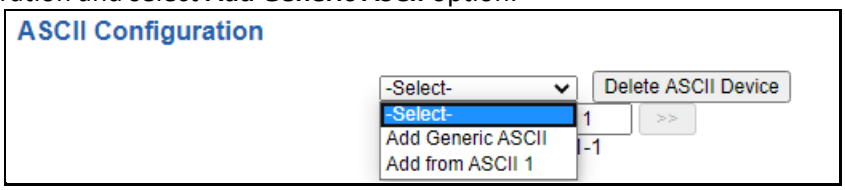
# ASCII Configuration

After the port configuration has been completed, click the **ASCII** button to continue configuration.

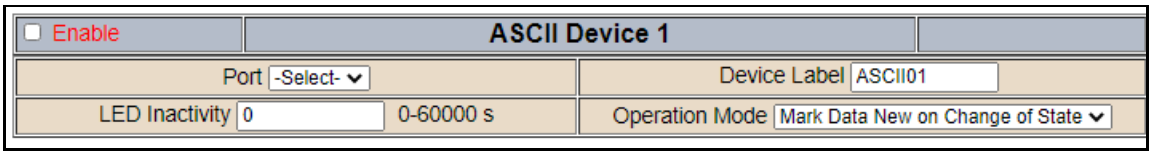



1) To add an ASCII device or additional ASCII device, click the -Select- dropdown menu under ASCII Configuration and select **Add Generic ASCII** option.

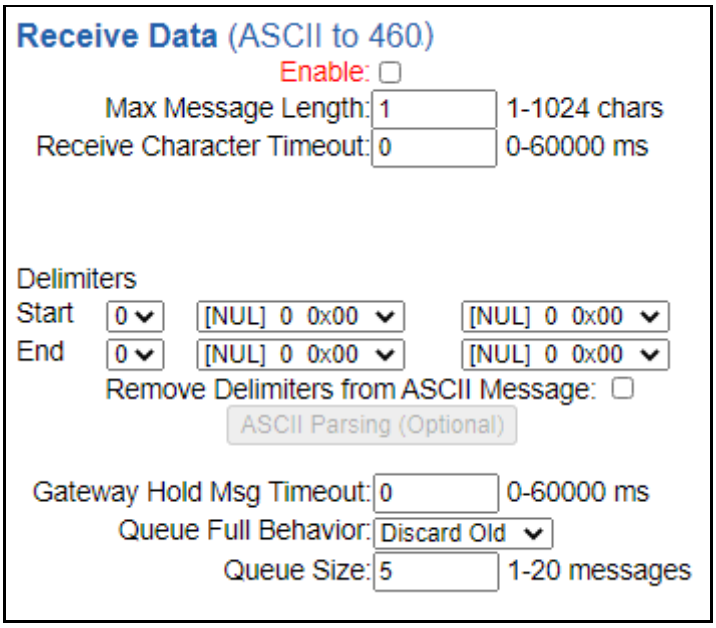



   a) To remove a device, navigate to the ASCII device to delete using the **<<** and **>>** buttons and click the **Delete ASCII Device** button.
   b) To create a new ASCII device with the same parameters already configured from another ASCII device, click the -Select- dropdown menu and select the **Add from ASCII X** option (where X represents the ASCII device you wish to copy parameters from).
   c) Once created, you can make any additional changes needed to that new ASCII device.
2) The **Enable** check box should be selected for the device.
3) **Port**: Select which port is being used for communication. This port must be configured on the Port/USB or TCP/IP (depending on your product) Configuration page. If it has not yet been configured, it will not display in this dropdown.
4) Enter a **Device Label** to identify the device within the gateway.
5) **LED Inactivity Timeout**: Enter the amount of time, in seconds, to wait before flashing the LED red indicating that no messages have been received or transmitted during this time.
6) **Operation Mode:**
   a. Mark Data New on Change of State: Send data to the mating technology, on a per point basis, upon a change of state. For more explanation see the Receive Data – Operation Mode section below.
   b. Mark Data New on New Message: Send data to the mating technology for all data points, no matter change of state or not. For more explanation see the Receive Data – Operation Mode section below.



## Receive Data

This side is configured to receive data from the ASCII device into the gateway.




Use the following fields to determine when a message has been received.

1) **Enable**: Check this box to configure the Receive Data section.
2) **Max Message Length**: Enter the max number of characters that can be received by the gateway.

   *Example:* Max Message Length is set to 5 and the message of "helloworld" was sent by the ASCII device. This will be sent to the other protocol as "hello" followed by "world" as two messages.
3) **Receive Character Timeout:** Enter the max amount of time (in ms) between characters that the gateway will wait before issuing a timeout and calling the message complete.

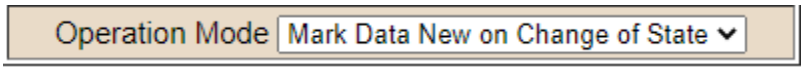
   *Example:* If Max Message Length varies in size, then use a timeout to call the message complete rather than message length. You can also use End Delimiters (below) to call a message complete.
4) **Number of Start Delimiters:** Select the number of delimiters that the gateway should look for before the gateway processes the data.
5) **Select Start Delimiters:** Select the start delimiters that the gateway should look for.
6) **Number of End Delimiters:** Select the number of delimiters that the gateway should look for to call a message complete.
7) **Select End Delimiters:** Select the end delimiters that the gateway should look for.
8) **Remove Delimiters from ASCII Message:** If checked, the gateway will remove all delimiters that have been configured before sending it to the other protocol.
9) **ASCII Parsing (Optional):** Additional parsing can be performed on the string before being passed to the other protocol. See the ASCII Configuration - ASCII Parsing section for more information.

10) **Gateway Hold Msg Timeout:** Enter the amount of time (in ms) to wait before sending a new message to the other protocol.
11) **Queue Full Behavior**: Select which message to discard when the queue is full. Once the queue is full, the gateway will discard either the oldest or newest message (Only used if Gateway Hold Msg Timeout is non-zero).
12) **Queue Size**: Select how many complete messages the gateway will hold before starting to discard (Only used if Gateway Hold Msg Timeout is non-zero).

## Receive Data – Operation Mode

**Mark Data New on Change of State (COS)**

When data comes into the RTA gateway, then it will be sent over to the matting protocol only if the data has a different value.

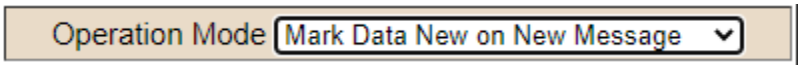



*Example for 460ETCA*

Operator scans "HelloWorld" with a barcode scanner. That data is gathered in the ASCII side of the RTA gateway, and is then processed and sent over to the ETC side and written over to the Allen-Bradley PLC. The next time the operator scans the same barcode "HelloWorld", the ASCII side gathers the data, but the data didn't change so it will not be sent over to the ETC portion of the RTA gateway. If the operator scans "1234567890" with the barcode scanner, the ASCII side of the RTA gateway will process the data. Since the data has changed it will be sent over to ETC and sent over to the PLC.

**Mark Data New on New Message**

When data comes into the RTA gateway, then it will be sent over to the matting protocol regardless if it's the same data. This allows you to send the same data over again to the mating protocol.

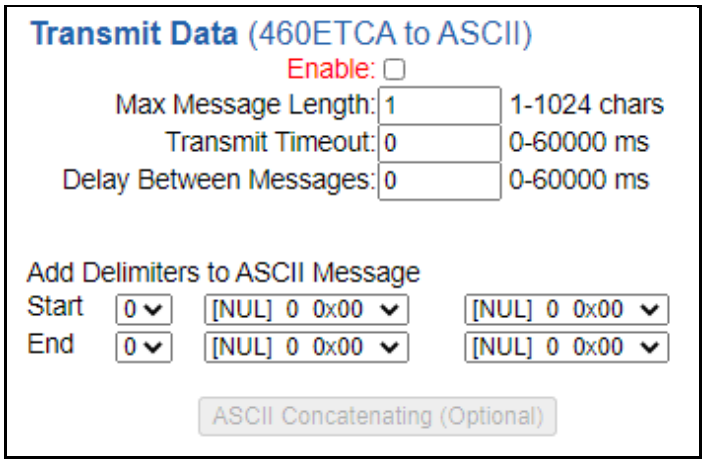



*Example ETCA*

Operator scans "HelloWorld" with a barcode scanner. That data is gathered in the ASCII side of the RTA gateway, and is then processed and sent over to the ETC side and written over to the Allen-Bradley PLC. The next time the operator scans the same barcode "HelloWorld", the ASCII side gathers the data, processes it, then sends it over to the ETC portion of the RTA gateway to be sent out. If the operator scans "1234567890", the ASCII side of the RTA gateway will process the data and send it over to ETC side and then to the PLC.

## Transmit Data

This side is configured to transmit data from the gateway into the ASCII device.




Use the following setup fields to help the 460 transmit an ASCII message.

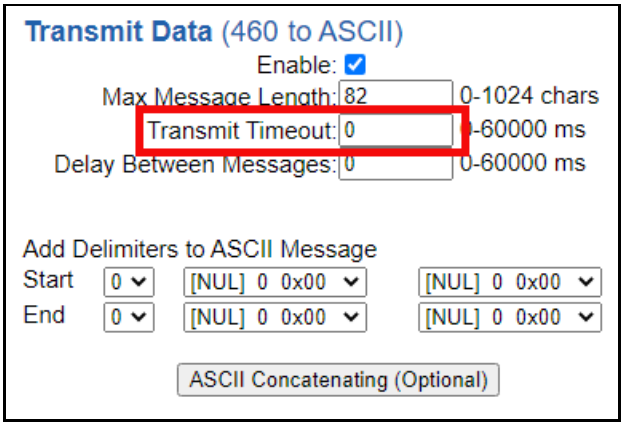
1) **Enable**: Check this box to configure the Transmit Data section.
2) **Max Message Length**: Enter the max number of characters that can be transmitted by the gateway.
3) **Transmit Timeout:** Enter the amount of time (in ms) that the gateway waits before sending an ASCII message (0 Sends Immediately). If the data has changed before the time expires, the gateway immediately sends the message to the ASCII device.
4) **Delay Between Messages:** Enter the amount of time (in ms) that the gateway waits before verifying a Change of State of the ASCII message OR will start the Transmit Timeout.
5) **Number of Start Delimiters:** Select the number of delimiters that will be added onto the beginning of the ASCII string.
6) **Select Start Delimiters:** Select the start delimiters that should be added to the ASCII string.
7) **Number of End Delimiters:** Select the number of delimiters that will be added onto the end of the ASCII string.
8) **Select End Delimiters:** Select the end delimiters that should be added to the ASCII string.
9) **ASCII Concatenating (Optional):** Additional concatenating can be performed on the string before being written to the ASCII device. See the ASCII Configuration – ASCII Concatenating section for more information.

## Transmit Data – Triggering Methods

There are 3 methods that determine when the message is ready to be transmitted to your ASCII device:

1) **Cyclic** – This means that every *x* ms a new ASCII message will be transmitted, regardless of whether the data has changed or not.
2) **Triggering** – This means that a trigger event determines when a new ASCII message will be transmitted.
   Some methods can co-exist with others. Here are the optional rules:
   a) **Option 1:** Change-Of-State is defaulted, so this method is chosen if the Transmit Timeout field is left at 0 and **ALL** data is new.

      Example 1: From the PLC, send a message of "hello", through the RTA gateway, the ASCII device sees "hello". If you send "hello' again, nothing will happen because of the RTA Change-of-State Rule.



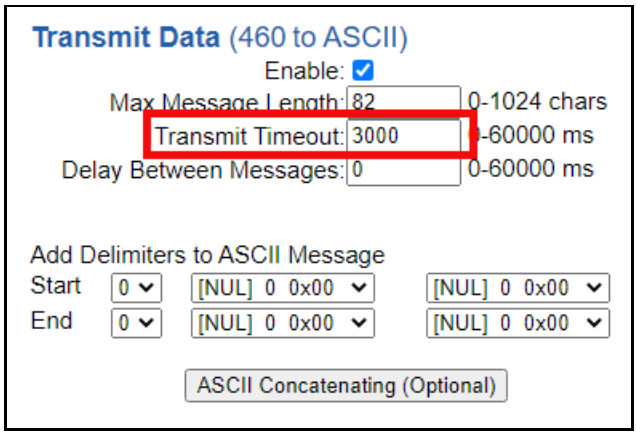

   b) **Option 2:** Technology Triggering (A/USB/TCP/WI). This method is chosen if the Transmit Timeout field is left at 0 and the trigger variables (as described in ASCII Configuration – Technology Triggering Method section of this manual) are mapped. This will disable Change-of-State. This method is recommended if your product in **NOT** an 460ETC product.

      **NOTE: If you have an 460ETC it's high recommended you use the Optimization Triggering.**

      *Example 2: Using the Technology Triggering Mappings (shown below in the Technology Triggering Method section*) you can make the data new only with a trigger. If you want to send the same/new message of based on a trigger and NOT cyclically, then keep the Transmit Timeout at 0 **AND** use the 2 Trigger Mappings. *See below for more examples in the* ASCII Configuration – Technology Triggering Method.

c. **Option 3:** Cyclic and Trigger can co-exist. For this to happen, the Transmit Timeout field needs be to nonzero and the trigger variables (as described in ASCII Configuration – ASCII Message Triggering Method section of this manual) are also mapped.

*Example 3*: From the PLC, send a message of "hello", based on whatever gets triggered first (the 3000ms Timeout or the trigger data point) the ASCII device will get updated. If the trigger data point is not updated, the then the RTA gateway will send the data every x ms to the ASCII device. See below for more examples in the ASCII Configuration – Technology Triggering Method.

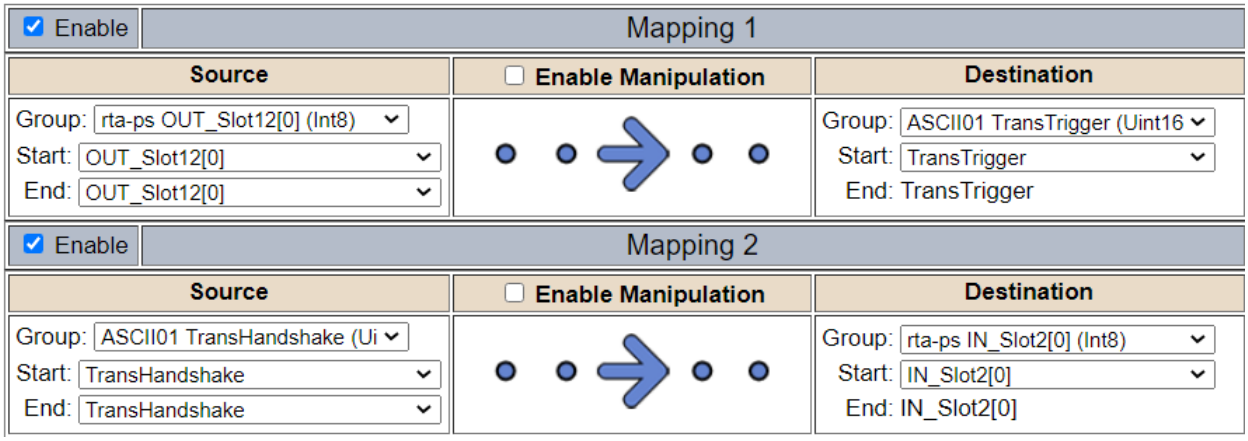


## ASCII Configuration – Technology Triggering Method

This method allows the other protocol to signal when to send the next message using data handshakes. These "signals" are controlled using data variables (TransTrigger and TransHandshake) already in the mapping. The method will send the new/old data when triggered.

**NOTE**: These two data variables will need to be mapped manually on the Data Mapping webpage since it will not be mapped using Auto-Map.

While these two data variables are mapped, the Change-of-State method is disabled but messages can still be sent via the cyclic method, if configured. For more information on the ASCII triggering methods, please see the Transmit Data – Triggering Methods section of this user guide.




**How the triggering method works: The example shown below is our 460PS* (*A/TCP/USB)**

1) The mating protocol sends a numbered value to the ASCII TransTrigger diagnostic variable. This value must be different from the previous value for a new message to be triggered. The following example is Slot 12[0] as the trigger for the PLC to update everything in Slot 11[0] which is the data.

2) Depending on the TransTigger value in the Display Data page, one of 4 things will occur:
   a) If TransTrigger = 65535, then the triggering method is disabled. Usually on powerup.

***RTA460 Display Data Example:***

| ASCII | | | 460PSA | Profinet IO | | |
|---|---|---|---|---|---|---|
| **Name** | **Value (Hex)** | | **Manipulation** | **Name** | **Value (Hex)** | |
| TransTrigger | 65535 | 0xFFFF | ← | OUT_Slot12[0] | -1 | 0xFF |

b) If TransTrigger = 0, then the triggering method is enabled, but no message will transmit.

***RTA460 Display Data Example:***

| ASCII | | | 460PSA | Profinet IO | | |
|---|---|---|---|---|---|---|
| **Name** | **Value (Hex)** | | **Manipulation** | **Name** | **Value (Hex)** | |
| TransTrigger | 0 | 0x0000 | ← | OUT_Slot12[0] | 0 | 0x00 |
| Trans_Field01 | 11 | 0x000B | ← | OUT_Slot11[0] | 11 | 0x0B |

***TIA Portal Example:*** The data will still go to the RTA gateway, however the RTA gateway will NOT transmit the data to the ASCII device until the Slot12[0] triggers the TransTrigger.

| | Name | Address | Display for.. | Monitor ... | Modify va... | | Comment |
|---|---|---|---|---|---|---|---|
| 1 | | %QB80 | DEC+/- | | 11 | ☑ ⚠ | Data to RTA ASCII |
| 2 | | %QB88 | DEC+/- | | 0 | ☑ ⚠ | Trigger data to RTA |

c) If TransTrigger is between 1-65534 **AND** the value **IS** equal to the TransHandshake diagnostic variable, then no new message will transmit, until Slot 12[0] triggers again.

***RTA460 Display Data Example:***

| ASCII | | | 460PSA | Profinet IO | | |
|---|---|---|---|---|---|---|
| **Name** | **Value (Hex)** | | **Manipulation** | **Name** | **Value (Hex)** | |
| TransTrigger | 1 | 0x0001 | ← | OUT_Slot12[0] | 1 | 0x01 |

| ASCII | | | 460PSA | Profinet IO | | |
|---|---|---|---|---|---|---|
| **Name** | **Value (Hex)** | | **Manipulation** | **Name** | **Value (Hex)** | |
| TransHandshake | 1 | 0x0001 | → | IN_Slot2[0] | 1 | 0x01 |

d) If TransTrigger is between 1-65534 **AND** the value **IS NOT** equal to the TransHandshake diagnostic variable, then a new message will be transmitted. The value in TransTrigger will then be moved to TransHandshake.

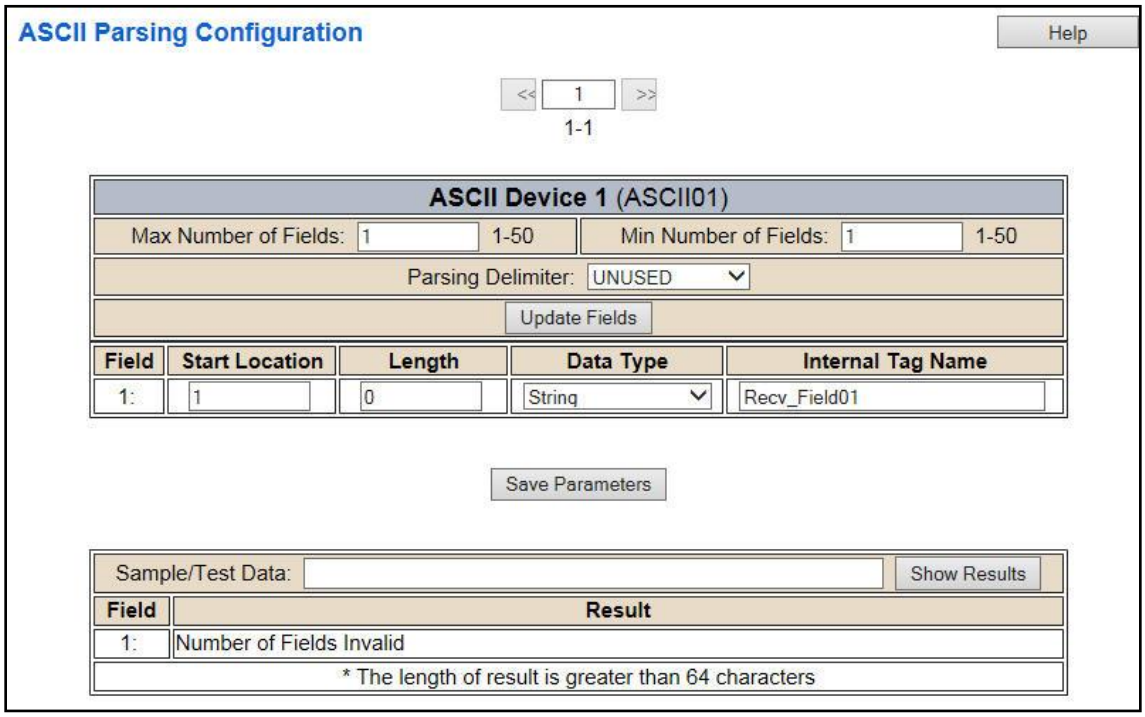
***TIA Portal Example:*** Once the Slot12[0] increments (data is sent from the PLC to the ASCI device) then the Slot2[0] will get updated with the handshake

| | Name | Address | Display format | Monitor value | Modify value | | Comment |
|---|---|---|---|---|---|---|---|
| 1 | | %QB80 | DEC+/- | | 11 | ☑ ⚠ | Data to RTA ASCII |
| 2 | | %QB88 | DEC+/- | | 1 | ☑ ⚠ | Trigger data to RTA |

| | Address | Display format | Monitor value | Modify value | | Comment |
|---|---|---|---|---|---|---|
| 1 | %IB92 | DEC+/- | 0 | | ☐ | Data From RTA |
| 2 | %IB100 | DEC+/- | 1 | | ☐ | RTA Handshake |

## ASCII Configuration – ASCII Parsing

The ASCII parsing feature allows you to break apart an incoming ASCII string by delimiter or character offset into multiple data fields. You can then apply a data type to the fields and deliver them to user defined locations in the mating protocol. Click the **ASCII Parsing (Optional)** button at the bottom of the ACII Configuration page to access the ASCII Parsing Configuration page for this device.

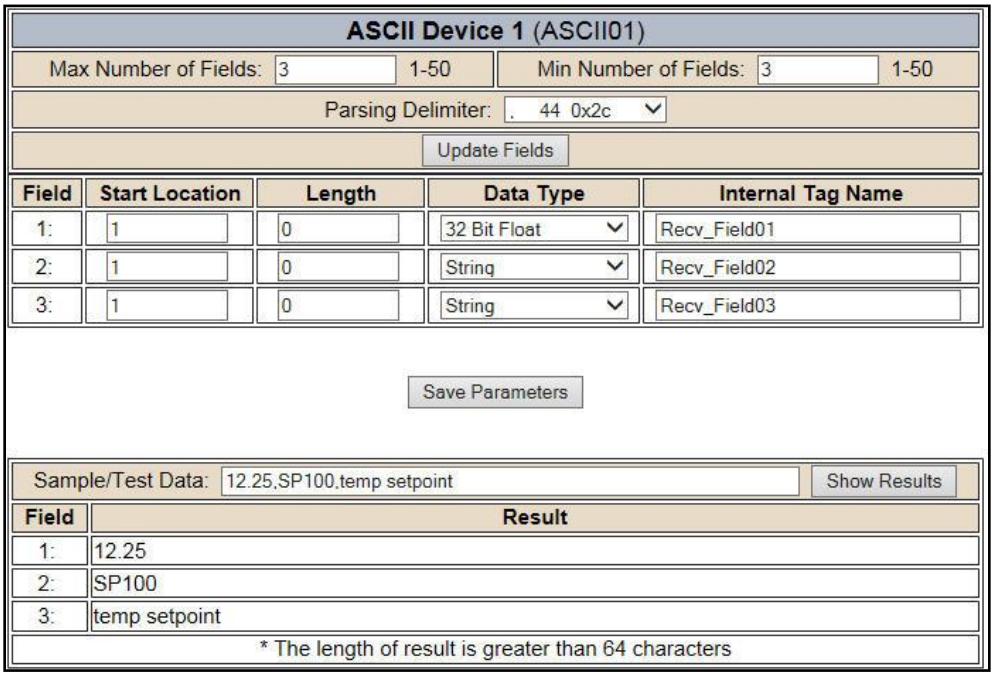



1) **Max Number of Fields:** This indicates the max number of fields the ASCII data will be parsed into (up to 50 values per message).
2) **Min Number of Fields:** This indicates the min number of fields that must be present in an ASCII string for the message to be considered valid. An error will be flagged if the actual number of fields is less than this value.
3) **Parsing Delimiter:** This defines the delimiter that will be used to parse an ASCII message. If delimiters are not present, select UNUSED and use the length fields to parse the message.
4) **Start Location & Length:**
    a. If a parsing delimiter is used, the **Start Location** will be the first character of the data field. The **Length** will be the number of characters from the Start Location. If the **Length** is 0, the gateway will read the entire field.
    b. If the parsing delimiter is unused, then the **Start Location** will be the first character of the string. The **Length** will be the number of characters from the Start Location. If the **Length** is 0, the gateway will read the entire message from the **Start Location** to the end of the ASCII string.
5) **Data Type:** Select the data type of the parsed value.
6) **Internal Tag Name:** Enter a name to reference this tag within the gateway's display and mapping pages.

## ASCII Configuration – ASCII Parsing Examples

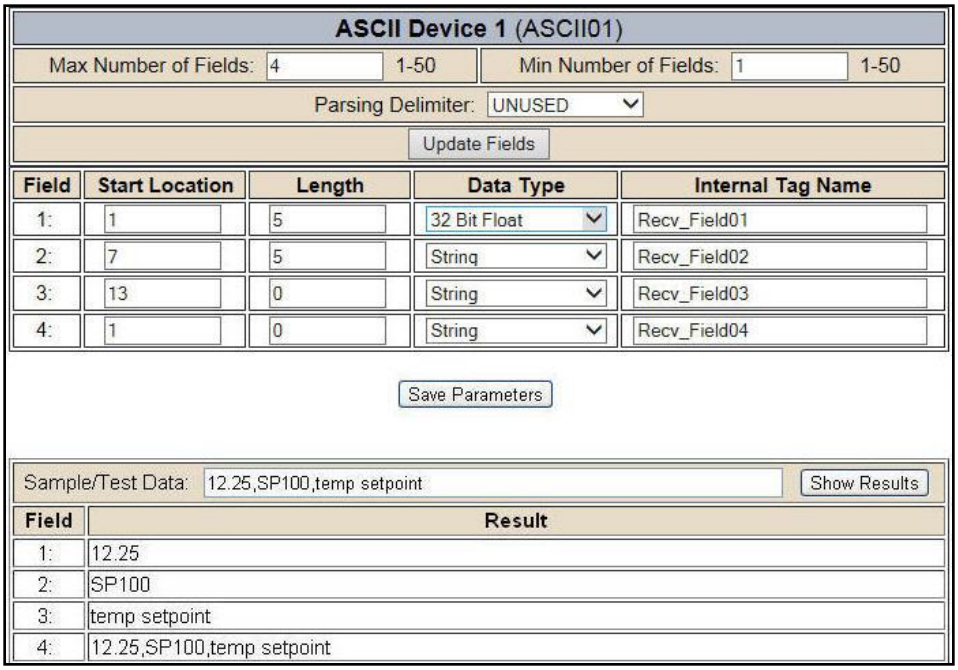
**Example #1 - Parsing a message using the Parsing Delimiter option:**

In this example, we are separating the string "12.25,SP100,temp setpoint" by a comma delimiter. The first value is being parsed into a float data type, the second and third values are being parsed into a string data type. Since the Min Number of Fields is 3, all 3 fields must be present for the message to be considered valid and processed. The output is seen below:




**Example #2 - Parsing a message without the Parsing Delimiter option:**

In this example, we are separating the fields in the string "12.25,SP100,temp setpoint" using the start and length parameters. The first value is being parsed from the 1st character for a length of 5 and stored into a float data type. The second value is being parsed from the 7th character for a length of 5 characters and stored into a string data type. The third value is being parsed starting from the 13th character for the rest of the remaining characters and stored into a string. The fourth value contains the entire ASCII message and is stored into a string. Only the first field needs to be present for the data to be considered valid and will be processed. If less than field 1 is present, the message will not be parsed and will be flagged an error. The output is seen below:



**ASCII Device 1** (ASCII01)

Max Number of Fields: 4 1-50 | Min Number of Fields: 1 1-50

Parsing Delimiter: UNUSED

Update Fields

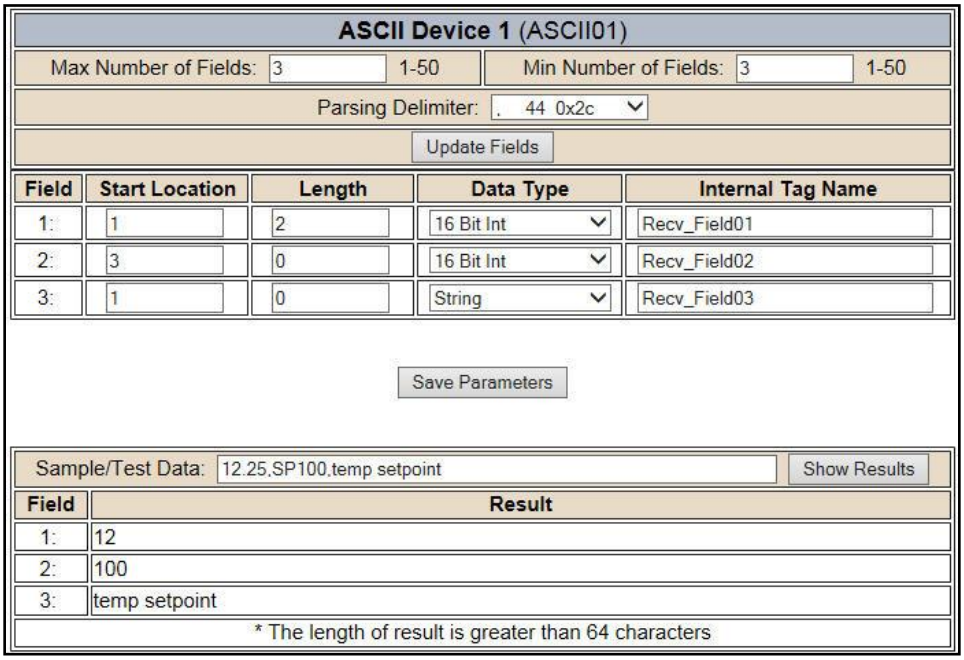
| Field | Start Location | Length | Data Type | Internal Tag Name |
|---|---|---|---|---|
| 1: | 1 | 5 | 32 Bit Float | Recv_Field01 |
| 2: | 7 | 5 | String | Recv_Field02 |
| 3: | 13 | 0 | String | Recv_Field03 |
| 4: | 1 | 0 | String | Recv_Field04 |

Save Parameters

Sample/Test Data: 12.25,SP100,temp setpoint Show Results

| Field | Result |
|---|---|
| 1: | 12.25 |
| 2: | SP100 |
| 3: | temp setpoint |
| 4: | 12.25,SP100,temp setpoint |

**Example #3 - Parsing a message using the Parsing Delimiter option and Start Location and Length:**
In this example, we are separating the fields in the string "12.25,SP100,temp setpoint" using the comma delimiter, the start, and length fields. The first value is being parsed from the 1st character for a length of 2 and stored into an integer data type. The second value is being parsed from the 3rd character of the second comma-parsed field for the remainder of that field and stored into an integer data type. The third value is being parsed starting from the 1st character of the third comma-parsed field for that entire field and stored into a string. All 3 fields need to be present for the message to be valid. The output is seen below:



**ASCII Device 1** (ASCII01)

Max Number of Fields: 3 1-50 | Min Number of Fields: 3 1-50

Parsing Delimiter: , 44 0x2c

Update Fields

| Field | Start Location | Length | Data Type | Internal Tag Name |
|---|---|---|---|---|
| 1: | 1 | 2 | 16 Bit Int | Recv_Field01 |
| 2: | 3 | 0 | 16 Bit Int | Recv_Field02 |
| 3: | 1 | 0 | String | Recv_Field03 |

Save Parameters

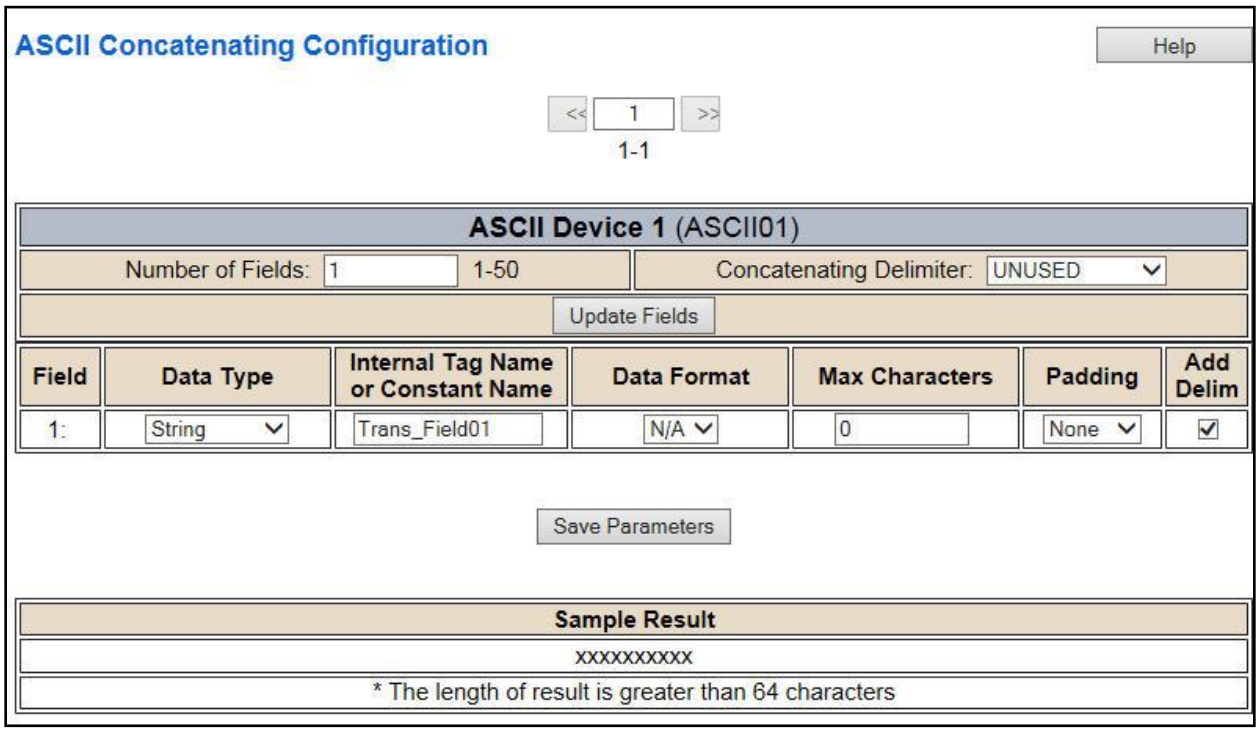
Sample/Test Data: 12.25,SP100,temp setpoint Show Results

| Field | Result |
|---|---|
| 1: | 12 |
| 2: | 100 |
| 3: | temp setpoint |

* The length of result is greater than 64 characters

## ASCII Configuration – ASCII Concatenating

The ASCII concatenating feature allows you to combine multiple data points and locations, in the mating protocol, into a single ASCII string. Click the **ASCII Concatenating (Optional)** button at the bottom of the ACII Configuration page to access the ASCII Concatenating Configuration page for this device.




1) **Number of Fields:** This indicates how many values will be concatenated together to form a single ASCII message (up to 50 values per message).
2) **Concatenating Delimiter:** This adds a delimiter between data fields in the ASCII string. If a delimiter should not appear between each of the fields, select UNUSED.
3) **Data Type:** Select the data type of the parsed value.
    a. Signed and Unsigned 8/16/32/64 Bit Integers
    b. 32/64 Bit Floating Points
    c. String – in order to use, a String data type must be selected in the other protocol. Cannot concatenate an Integer to a String.
    d. Constant String
4) **Internal Tag Name/Constant Name:**
    a. If data type other than Constant String is selected, then this will be the name to reference this tag within the gateway. This value is used on the display page and the mapping page.
    b. If data type Constant String is selected, then this is the string value that will send.
5) **Data Format:**
    a. %d – used for Signed Integers
    b. %u – used for Unsigned Integers
    c. %lf – used for Floating Points with no set decimal precision
    d. %.1lf…%.6lf – used for Floating Points to show the offset of the decimal point value

i. EX: 123.456789 set as %.3lf will display as 123.456

e. %e – used for Exponential Notation
f. %x – used to represent Hexadecimal values for Signed/Unsigned Integers or Floating points
g. String and Constant String data types do not use this field

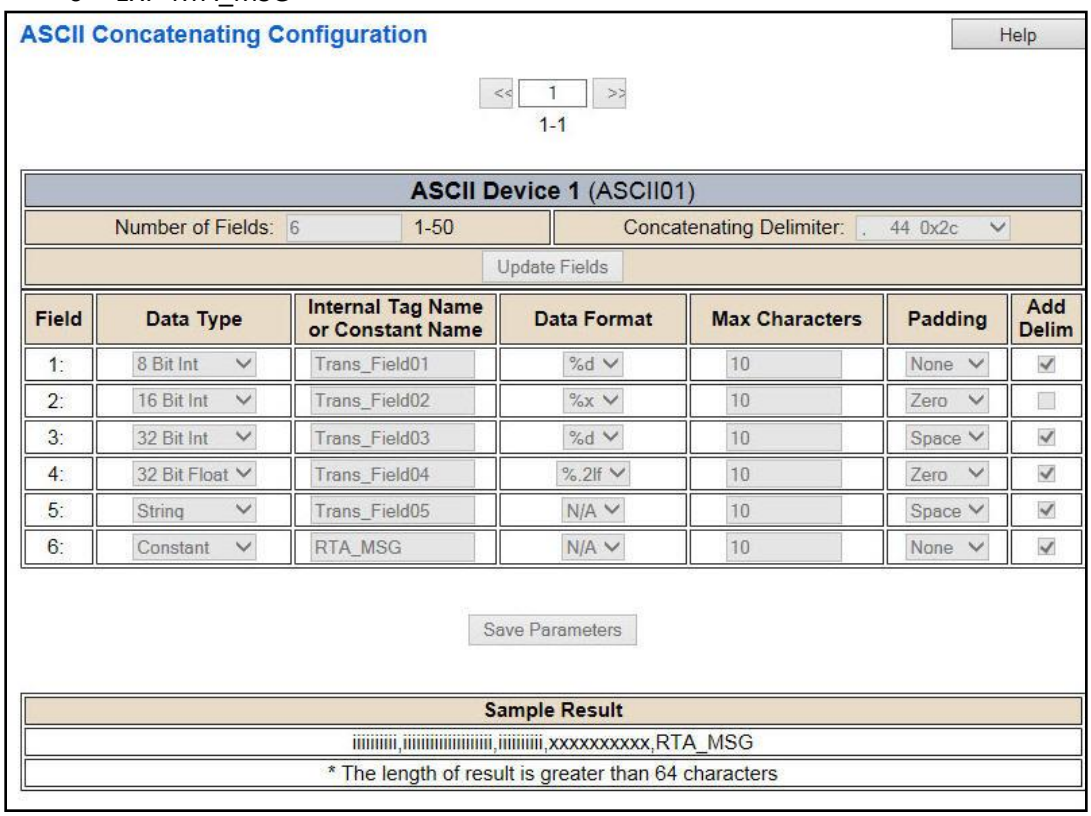
6) **Max Characters:** This is the max number of characters that can be transmitted for a single field.
Special Cases
   a. If set to 0, the entire field is transmitted.
   b. If the length of the value is less than the Max Characters, then the Padding Character will be used (if set).
   c. If the length of the value is greater than the Max Characters, then the value will be truncated.
7) **Padding:** If the length of the value is less than the Max Characters padding Zeroes, Spaces, or Nothing to the remaining character placeholders, the padding will occur to the left of the value.
8) **Add Delim:** Used when a concatenating delimiter is selected. Check to add the concatenating delimiter to the end of that field.
9) **Sample Result:** This will display an example of how the data will output. This will not display live data. It provides an example of the string structure.
**NOTE:** Sample Result field will only show the first 64 characters of the message.
   a. String data and constant data types will display as *x*'s.
   b. Any other data type will display as *i*'s.
   **NOTE**: For display purposes, if Max Characters is set to 0, only 10 characters will display for that field in the Sample Result section. The true value, if larger, will be processed correctly.
   EX: Field 1 is set for a String data type and Max Characters is set to 0, only 10 *x*'s will display in the sample result even though the max character length is set to 50.

# ASCII Configuration – ASCII Concatenating Examples

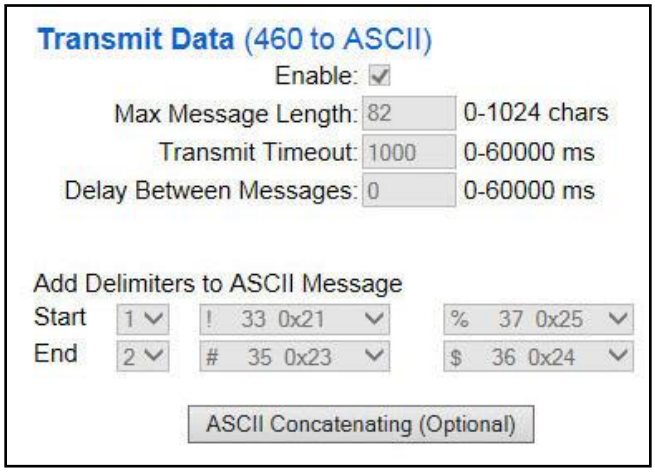
**Example #1 - Concatenating a message using the concatenating delimiter option:**

In this example, the comma is selected as the concatenating delimiter. Let's look at each field closer:

1) Field 1 –8 bit int represented as Trans_Field01 in the gateway. It will output as an integer with a max of 10 characters. No padding is used and a comma will be added to the end of the value.
   - EX: "34,"
2) Field 2 –16 bit int represented as Trans_Field02. It will output in Hexadecimal with a max of 10 characters, padded with zeros and no comma will be added to the end of the value.
   - EX: "00000000A0"
3) Field 3 –32 bit int represented as Trans_Field03. It will output as an integer with a max of 10 characters, padded with spaces and a comma will be added to the end of the value.
   - EX: "____123456," (shown with _'s to see spaces)
4) Field 4 –32 bit float represented as Trans_Field04. It will output as a float with 2 decimal places with a max of 10 characters, padded with zeros and a comma will be added to the end of the value.
   - EX: "00001234.56,"
5) Field 5 –String represented as Trans_Field05 in the gateway. It will output as string with a max of 10 characters, padded with spaces and a comma will be added to the end of the value.
   - EX: "___testing," (shown with _'s to see spaces)
6) Field 6 – Constant String will output as "RTA_MSG" with a max of 10 characters. No padding is used and no comma will be added to the end (though checked) since it is the last field.
   - EX: "RTA_MSG"



Below is a sample of the Transmit Data is set up with the following delimiters:




**Example 1 Sample Result:** This use case is sending data via 5 PLC tags. Using the concatenating setup example and the transmit example, the ASCII data will be displayed within your ASCII device shown in the example below.

```
!123,0000003039      1234,0000123.45,ASCII Test,RTA_MSG#$
```

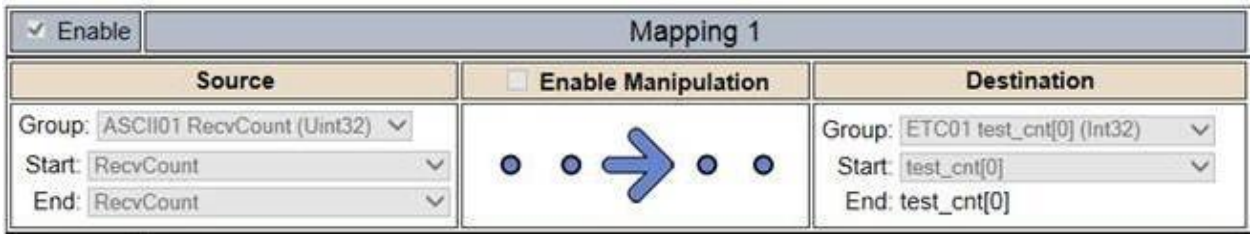
## ASCII Configuration – ASCII Message Counter

There is an additional ASCII variable that is very useful to access within the gateway's mating protocol. This data variable will need to be added manually since it will not be mapped using Auto-Map.

**RecvCount**- indicates how many ASCII messages have been successfully read by the gateway for that device. A successful incoming message means that at least one of our three end cases (Max Length, Timeout or Delimiters) has been met. This will match the Diagnostic Variable Successful Receive Count for each ASCII device.

This variable can be mapped to the mating protocol using the Data Mapping webpage. It is mapped just like the Status_XY variable described in the Data Mapping- Adding Diagnostic Information section of this user guide.

**Example**: For this example, the other protocol in the gateway is the Allen-Bradley PLC. As you can see from the picture below, the RecvCount for ASCII Device 1 is mapped to the first index of a PLC tag array called test_cnt. The data type of this tag is an Int32 to match the data type of RecvCount. The tag test_cnt[0] will now hold the number of successfully read messages from ASCII Device 1.

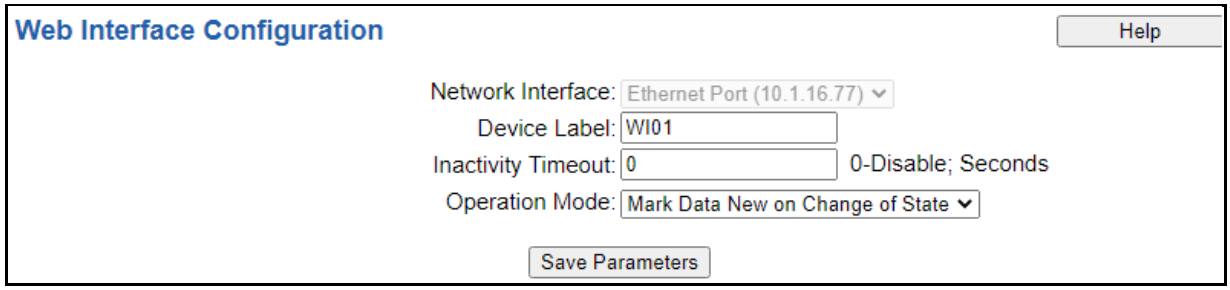



**Application Use:** This is particularly useful for applications connecting devices like barcode scanners and weigh scales. The gateway will cyclically update the mating protocol with the last ASCII message sent, a change in the RecvCount is the only way to identify a new message if the messages are identical.

## Web Interface Configuration

Click the **Web Interface** button to access the configuration page.

1) Select which **Network Interface** to use for the web interface.
2) Enter a **Device Label** to identify the device within the gateway.
3) **Inactivity Timeout**: If the gateway has not received any messages within the entered time interval, in seconds, then the gateway will change the status of its connection to be "Idle". Enter a value of zero to disable this feature.
4) **Operation Mode:**
   d. Mark Data New on Change of State: Send data to the mating technology, on a per point basis, upon a change of state. For more explanation see the Operation Mode section below.
   e. Mark Data New on New Message: Send data to the mating technology for all data points, no matter change of state or not. For more explanation see the Operation Mode section below.
5) There are two different modes in which the Web Interface may operate in the gateway:
   a) Sever Mode
   b) Client Mode




## Web Interface Gateway Server Configuration

1) Server Mode – where the gateway acts as a web server and responds to HTTP POST and HTTP Get requests from a web services client, such as a web browser or Excel.
   a. No special configuration is needed to configure the server side of the gateway. Skip to the Web Interface Data Point List Configuration section of this manual and set up the data points.

## Web Interface Gateway Client Configuration

1) Client Mode – is where the gateway acts as a web client and initiates a connection to a web server, such as Apache or IIS (Internet Information Services). The gateway would then send all configured data points to the web server cyclically in a HTTP POST operation.
   a. To configure the client side of the gateway, configure the Automatic Data Transfer to User Host section. Then proceed with the rest of the Web Interface Data Point List Configuration section of this manual.
   b. The gateway is always enabled as a server and may act as both a server and client at the same time.

## Automatic Data Transfer to User Host (Used as a Web Client)

This section configures the gateway when it is operating as a web client, uploading XML/JSON data cyclically to an external, user-operated web server. This feature is optional.

**Disable/Enable:** If set to "Disable", the gateway will operate only as a web server and will not initiate a connection with an external user-operated web server. If set to "Enable", an optional proxy configuration section also may be configured.

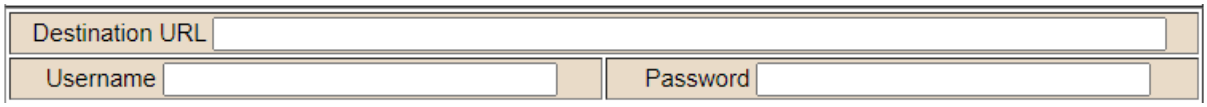
1) **Destination URL:** Enter the web address that the gateway will POST XML/JSON data to. Addresses should begin with the protocol designation, such as **Error! Hyperlink reference not valid.**.

   **NOTE:** If the web server uses a non-standard port number, the port should be included in the URL. Examples are:

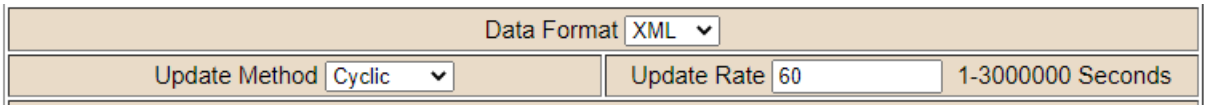
   a. *http://yourserver.yourdomain/upload.php*
   b. *http://192.168.100.1:8080* (if special port 8080 is to be used)
   c. *http://yourserver.yourdomain:8080/upload.asp* (if special port 8080 is to be used)

2) **Username/Password**: If the Destination URL entered previously requires basic HTTP authentication, then the username or username/password should be entered here.

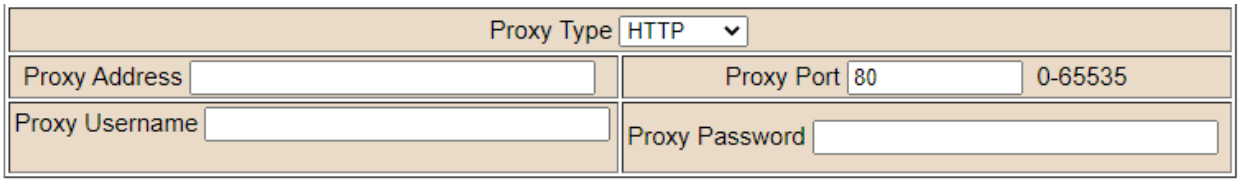
   **NOTE**: Basic HTTP Authentication is subject to interception by a third party on an improperly secured network.




3) **Data Format**: Select whether the file should be in XML or JSON format.
4) **Update Method**: Select whether to write the data Cyclically, Triggering, or Both
5) **Update Rate**: Enter a time, in seconds, between cyclic uploads to the Destination URL.




6) **Proxy Type**: Select the option that describes if any special configuration is required to access the web server:
   a. **None** – No proxy or a transparent proxy is being used and the subsequent fields in this table are disabled.
   b. **HTTP** – The web server is behind a Hypertext Transfer Protocol (HTTP) Proxy Server.
   c. **SOCKS5** – The web server is behind a Socket Secure version 5 (SOCKS5) Proxy Server.



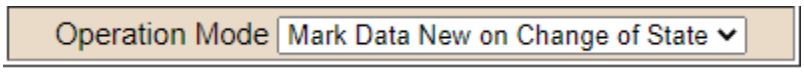
**Proxy Address**: If a proxy is selected, enter the address of the proxy. This can either be a hostname such as *proxy.yourcompany.com* or a plain IP address such as *192.168.100.1*.

7) **Proxy Port**: If a proxy is selected, enter the TCP port that the proxy will use. Default port for HTTP proxy is 80 and 1080 for SOCKS5.
8) **Proxy Username/Proxy Password**: If a proxy is selected and requires authentication, enter the username or username/password here.

## Operation Mode

**Mark Data New on Change of State (COS)**

When data comes into the RTA gateway, it will be sent over to the matting protocol only if the data has a different value.

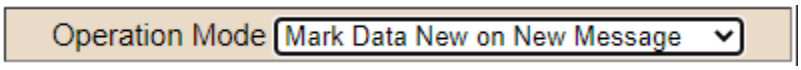



*Example for 460ETCWI*

Operator sends "HelloWorld" from the PLC. That data is gathered in the WI side of the RTA gateway and is processed and sent over to the web sever. Next time the operator sends the same data "HelloWorld". The WI side gathers the data, but the data didn't change so it will not be sent over to the WI portion of the RTA gateway. The operator sends "1234567890" from the PLC. The WI side of the RTA gateway will process the data and since the data has changed, it will be sent over to the web server.

**Mark Data New on New Message**

When data comes into the RTA gateway, it will be sent over to the matting protocol regardless if it's the same data. This allow you to send the same data over again to the mating protocol.




*Example for ETCWI*

Operator sends "HelloWorld" from the PLC. That data is gathered in the WI side of the RTA gateway and is processed and sent over to the web server. Next time the operator sends the same data "HelloWorld", the WI side gathers the data, processes it, then sends over to the web server. The operator sends "1234567890", the WI side of the RTA gateway will process the data and send it over to the web server.

## Web Interface Data Point List Configuration

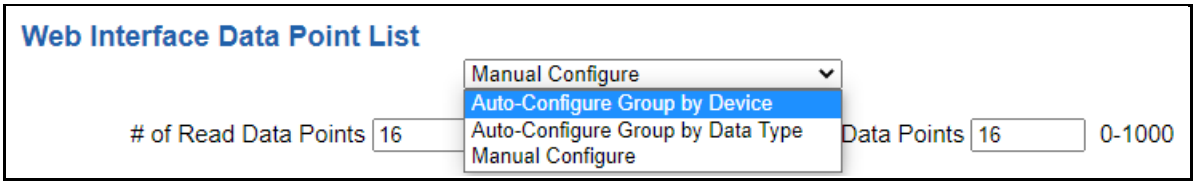
The bottom area of the Web Interface Configuration page allows configuration of 1000 data points in each direction.

> **NOTE**: Due to the limited number of internal mappings inside the gateway, though each direction in the web interface may be configured for up to 1000 data points, there is an overall limit, between the read and write direction, of 1000 data points total.

There are three ways to configure this protocol:

1) Auto-Configure Group by Device (Default)
2) Auto-Configure Group by Data Type
3) Manual Configure

**NOTE**: You may go back and forth between modes, but when reverting from Manual Mode to either of the two Auto-Configure Modes, all changes made in Manual Mode will be discarded.



## Web Interface Data Point List Configuration: Auto-Configure (Default)

While in either of the two Auto-Configure Modes, the number of data points and the actual data points themselves cannot be edited. Auto-Configure Mode looks at the other protocol and then configures the data point list within the web interface to match. The web interface names and types will be defined after the other protocol is configured.

The data will be configured according to the following rules:

1) Any Coil or 1 Bit Binary Pack data will be mapped as **Bool.**
2) Any 8 Bit Binary Pack data will be mapped as **Bitpack (8 Bits).**
3) Any 16 Bit Binary Pack data will be mapped as **Bitpack (16 Bits).**
4) Any 32 Bit Binary Pack data will be mapped as **Bitpack (32 Bits).**
5) Any 8 Bit Int data will be mapped as **INT (8 bit)**.
6) Any 16 Bit Int data will be mapped as **INT (16 bit)**.
7) Any 32 Bit Int data will be mapped as **INT (32 bit)**.
8) Any 64 Bit Int data will be mapped as **INT (64 bit)**.
9) Any 8 Bit Unsigned Int data will be mapped as **UINT (8 bit)**.
10) Any 16 Bit Unsigned Int data will be mapped as **UINT (16 bit)**.
11) Any 32 Bit Unsigned Int data will be mapped as **UINT (32 bit)**.
12) Any 64 Bit Unsigned Int data will be mapped as **UINT (64 bit)**.
13) Any 32 Bit Float will be mapped as **Float (32 bit)**.
14) Any 64 Bit Float will be mapped as **Double (64 bit)**.
15) Any String Data Types will be mapped as S**tring**.
16) The read or write direction depends on whether it is configured as a read or write on the other protocol.
17) If the other protocol exceeds the number of data points supported, nothing will be mapped. You will see the # of Data Points remain at zero and the main page will display the following error:

a) To fix this error, simply decrease the amount of data you configured on the other protocol so that the max number of data points is not exceeded or call customer support to increase the limits.

To add additional or edit existing data points you will need to go into Manual Configure Mode.

## Auto-Configure Group by Device vs. Auto-Configure Group by Data Type

There are two different methods for Auto-Configure: *Group by Device* or *Group by Data Type*.

**NOTE**: When using Auto-Configure with the web interface, the difference between the two methods is subtle. The *# of Read Data Points* and the *# of Write Data Points* will be the same regardless of which Auto-Configure method is chosen. The only difference between the two methods is the way the data is ordered in the data table.

There are a couple of rules to keep in mind when using Auto-Configure Mode:

1) If the other protocol inside the gateway is a server, slave, or adapter protocol, then there are no differences between the Auto-Configure Modes.

## Group by Device (Default Method)

Group by Device goes through the other protocol on the gateway and auto-configures the data points on the web interface for all the data points on the other protocol's first device. After it finishes with the first device, it will auto-configure all the points for the second device (if one is configured), and so on.

## Group by Data Type

Group by Data Type goes through the other protocol on the gateway and automatically creates the number of data points in the web interface to match the total number of data points for each specific data type in the other protocol.

***Example***: *Protocol A is a master/client protocol that has two devices with the following setup:*

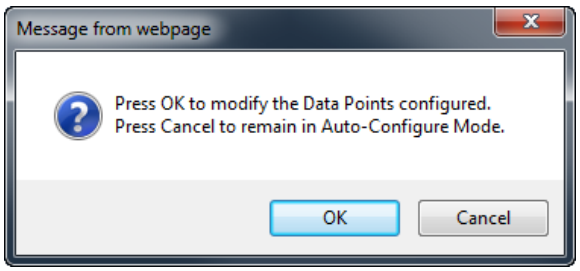
*Device_1 has 1 integer scan line, 1 float scan line, 1 integer scan line- each for 1 point of data*
*Device_2 has 1 integer scan line, 1 float scan line, each for 2 point2 of data*

*Protocol B is the Web Interface protocol that will be mapped with seven data points, but the ordering differs slightly between the two auto-configure modes:*

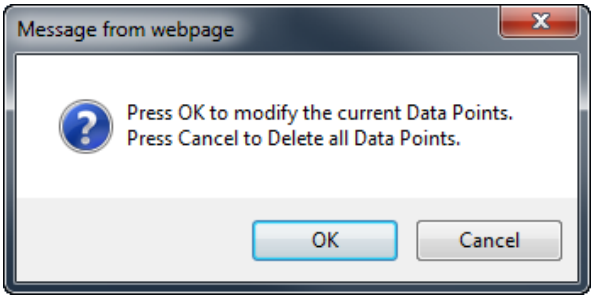
| Group by Device | Group by Data Type |
|---|---|
| Data Point 1 => Type Integer (DeviceA:Int1) | Data Point 1 => Type Integer (DeviceA:Int1) |
| Data Point 2 => Type Integer (DeviceA:Int1) | Data Point 2 => Type Integer (DeviceA:Int2) |
| Data Point 3 => Type Float (DeviceA:Float2) | Data Point 3 => Type Integer (DeviceB:Int1) |
| Data Point 4 => Type Integer (DeviceB:Int1) | Data Point 4 => Type Integer (DeviceB:Int2) |
| Data Point 5 => Type Integer (DeviceB:Int2) | Data Point 5 => Type Float (DeviceA:Float1) |
| Data Point 6 => Type Float (DeviceB:Float1) | Data Point 6 => Type Float (DeviceB:Float1) |
| Data Point 7 => Type Float (DeviceB:Float2) | Data Point 7 => Type Float (DeviceB:Float2) |

## Web Interface Data Point List Configuration: Manual Mode

1) To transition from either of the two Auto-Configure Modes to Manual Configure Mode, click the dropdown at the top of the Web Interface Configuration page and select Manual Configure.
   a) When prompted, click **OK** to confirm mode change or **Cancel** to remain in Auto-Configure Mode.



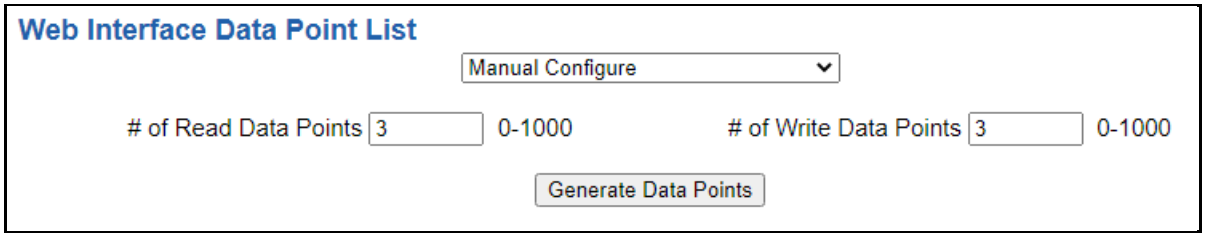

2) Once OK is clicked, there are two options on how to proceed:




3) To keep the data points that are already configured, press **OK**.
   a) You would want this option if you are adding additional data points or you want to modify the data point(s) that already exist.
4) To delete the data points that are already there and start over, press **Cancel**.

## Configuring Read and Write Data Points

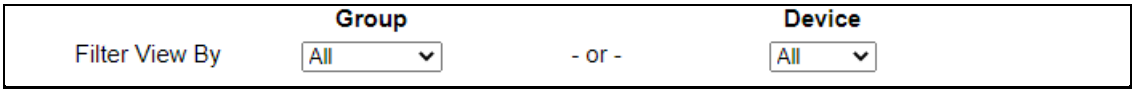
Follow these steps to manually configure read and write data points. Most of the time, the # of Read/Write Data Points number should be left at the value generated during the Auto-Configure method. However, if additional points need to be added, or manual configuration is preferred, enter the desired number of data points in these boxes.




1) **Generate Data Points:** Once values in the number of read/write data points has been changed, click this button to have them auto-generate. These new data points may be configured after they have been generated.

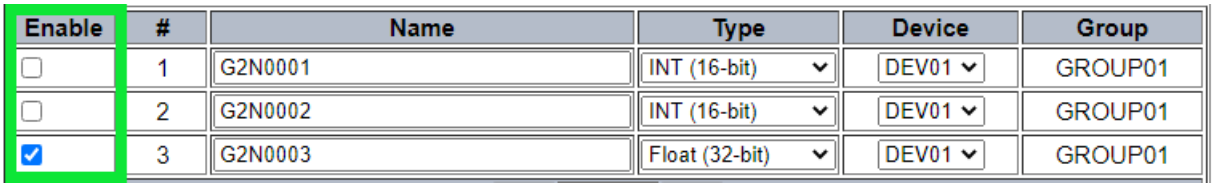
2) Select the **View Read Data Points** or **View Write Data Points** button.

1) The dropdowns next to **Filter View By** allow the data points table to be filtered so that only the data points assigned to a specific group or device are displayed. Reference the Web Interface Group Configuration for more explanation.




   a) To remove the applied filter, select *All* from the dropdown.
   b) Only one filter can be applied at a time.

2) To individually disable data points, uncheck the **Enable** checkbox. This will omit that data point from the generated XML/JSON.



| Enable | # | Name | Type | Device | Group |
|---|---|---|---|---|---|
| ☐ | 1 | G2N0001 | INT (16-bit) | DEV01 | GROUP01 |
| ☐ | 2 | G2N0002 | INT (16-bit) | DEV01 | GROUP01 |
| ☑ | 3 | G2N0003 | Float (32-bit) | DEV01 | GROUP01 |

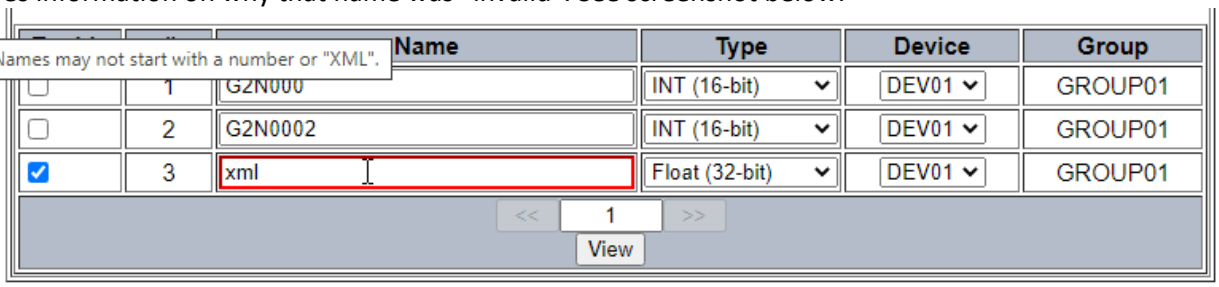
   **NOTE**: Attempts to write to a disabled write data point will generate an error in the gateway.

3) Enter a **Name** for the data point. This name will be used in the XML/JSON and must follow the following rules:
   a) Consist only of alphanumeric characters and the underscore character, no spaces.
   b) Length between 1 and 32 characters long
   c) Be unique among those assigned to a specific *Device*.
   d) Not begin with the string "XML".
   e) Reserved names of "DeviceName", "GroupName", "dataTimeStamp", and "Diagnostics" may not be used.

   **NOTE**: Although the data point names are treated as case-insensitive, the capitalization will be maintained in the generated XML/JSON as it was originally entered on this page.

4) After a name is entered, a verification check will run on this page to determine if the entered name is "valid". If a name is entered that fails one of the criteria described above, a popup message will appear.

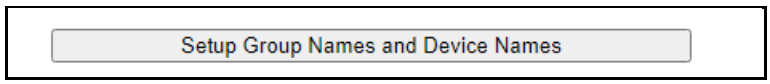
5) In addition, the name field will be highlighted in red and hovering the mouse on that error message gives information on why that name was "invalid". See screenshot below:




6) Select a **Type** for each data point. The Auto-Configure method will choose the best data type for that point, but the user may modify. This type should match the type of the data point on the other protocol that it will be mapped with.
7) Select a **Device** that this data point is associated with. This field is used, along with Group, to help organize the data in the XML/JSON. Up to 32 different devices can be configured. Each device is assigned to a **Group**. For more information and how to modify the name, please see the Web Interface Group Configuration page.
    i. If the other protocol is a client/master with multiple slaves/servers, typically this device field is used to represent each of those slaves/servers.
8) Click the **Save Parameters** button.

## Web Interface Group Configuration

Click the **Setup Group Names and Device Names** button at the bottom of the Web Interface Configuration page to access the Web Interface Group Configuration page.

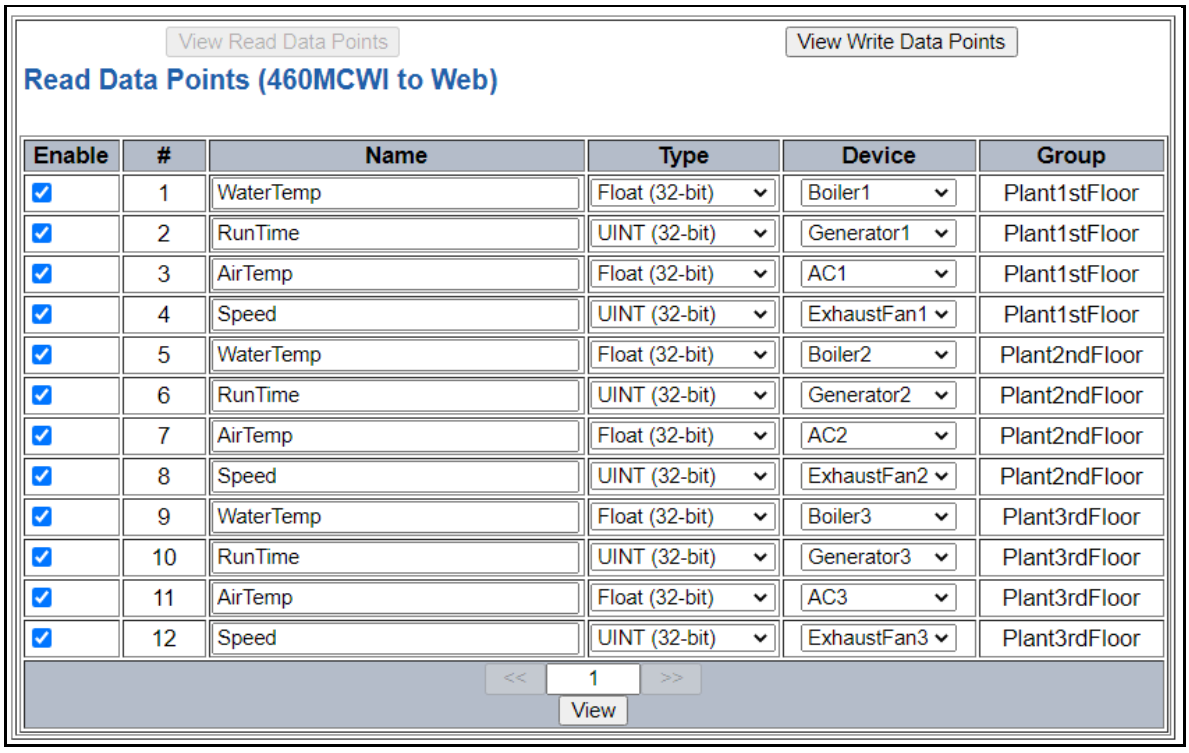



**NOTE**: When in Auto-Configure Mode, these fields are not configurable.

The Group Configuration page allows customization of each **Group Name** and **Device Name**. The gateway allows 16 possible groups. Within each group, there can be multiple devices assigned to a group. The **Group Associated To** column allows the device to be assigned to a group. The gateway allows a maximum of 32 devices. These names are subject to naming conventions that will be detailed below.

Below is an example of how someone would use this.

Let's say a building has 3 floors. You're looking to monitor water temp from all boilers, runtime from all generators, air temp from AC's and speed from the exhaust fans. The first image shows how you would configure the data points. Since we want to monitor 4 points on each floor, you'll need a total of 12 points. Once the Groups and Devices have been defined, come back to this page to assign a device to a data point name.




The second image displays how to setup the Groups and Devices. We are going to define floor numbers as a Groups and define all devices we are monitoring as displayed. Lastly, we can assign each device to a group. **Note: see below for naming rules.**

**Web Interface Group Configuration** Help

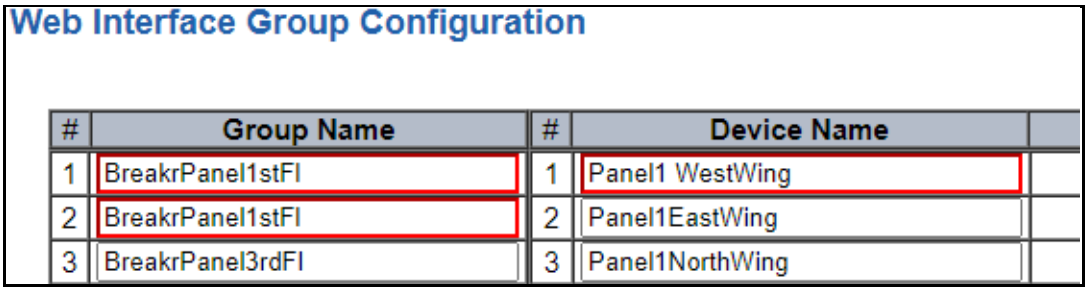
| # | Group Name | # | Device Name | Group Associated To |
|---|---|---|---|---|
| 1 | Plant1stFloor | 1 | Boiler1 | Plant1stFloor |
| 2 | Plant2ndFloor | 2 | Generator1 | Plant1stFloor |
| 3 | Plant3rdFloor | 3 | AC1 | Plant1stFloor |
| 4 | GROUP04 | 4 | ExhaustFan1 | Plant1stFloor |
| 5 | GROUP05 | 5 | Boiler2 | Plant2ndFloor |
| 6 | GROUP06 | 6 | Generator2 | Plant2ndFloor |
| 7 | GROUP07 | 7 | AC2 | Plant2ndFloor |
| 8 | GROUP08 | 8 | ExhaustFan2 | Plant2ndFloor |
| 9 | GROUP09 | 9 | Boiler3 | Plant3rdFloor |
| 10 | GROUP10 | 10 | Generator3 | Plant3rdFloor |
| 11 | GROUP11 | 11 | AC3 | Plant3rdFloor |
| 12 | GROUP12 | 12 | ExhaustFan3 | Plant3rdFloor |

## Requirements for Device and Group Names:

1) Name must be unique on this page.
   a) A group name may not also be a device name.
   b) No two group names may be the same.
   c) No two device names may be the same.
   d) Reserved names of “DeviceName”, “GroupName”, “dataTimeStamp”, and “Diagnostics” may not be used.
2) Name must be between 1-16 characters.
3) Only alphanumeric characters and the underscore character (‘_’) may be used, **No Spaces**.
4) Name must not start with a number.
5) Name must not start with the string “XML”.
   **NOTE**: Although the names are case-insensitive, the original capitalization will be maintained and used in the XML/JSON.

6) After a name is entered, a verification check will run on this page to determine if the entered name is “valid.” If a name is entered that fails one of the criteria described above, a popup message will appear.

7) In addition, the name field will be highlighted in red and hovering the mouse on that error message gives information on why that name was “invalid.” In the screenshot below, the Group name has duplicate names and the Device Name has a space in the first field.



Web Interface Group Configuration

| # | Group Name | # | Device Name |
|---|---|---|---|
| 1 | BreakrPanel1stFl | 1 | Panel1 WestWing |
| 2 | BreakrPanel1stFl | 2 | Panel1EastWing |
| 3 | BreakrPanel3rdFl | 3 | Panel1NorthWing |

8) When done making changes, click the **Save Parameters** button.

## Web Interface: XML/JSON Data Format

1) **XML** – For simplicity, the web interface uses a "flat" XML encoding format. The root XML element is always `<Devices>` followed by one or more `<Device>` elements, which will in turn contain name-value pairs with the data relevant to that device.
2) **JSON** – The format of JSON is very simplistic. Data is contained in a single array of zero or more objects, each representing a *device.* Each object contains one or more name/value pairs within the device.

## Retrieving Data from the Gateway from a Web Client

XML and JSON data accessed from the 460WI using web client is handled via a standard HTTP *GET* request. Which data is to be accessed and how that data is to be encoded is all determined by the URL supplied with the GET request. This is the same method through which web browsers retrieve web pages so you can use the web browser of your choice to experiment with different requests by simply typing the URL into the browser's address bar and hitting enter.

## Basics

To begin, you will need to know the IP address of your 460WI Gateway. If you do not know the IP address, refer to Accessing the Main Page Section. The examples in this document will use 192.168.0.1 for the sake of illustration, but your IP will likely be different.

The most basic GET request that can be sent to the 460WI encodes all of the data points configured in the Gateway. The only option in this case is deciding how you would like the data encoded.

If you would like the data encoded as XML, enter URL: http://192.168.0.1/gateway/xml

If you would like the data encoded as JSON, enter URL: http://192.168.0.1/gateway/json

**NOTE:** Use http://IP_ADDRESS_OF_YOUR_UNIT/gateway/xml or http://IP_ADDRESS_OF_YOUR_UNIT/gateway/json

All data requests will begin with one of the above URL paths. Further narrowing down your data requests involves adding additional information to the URL's path.

**Note:** A common mistake is to include an extra forward slash with the request for all the data in the Gateway (eg. `http://192.168.0.1/gateway/xml/` rather than the correct `http://192.168.0.1/gateway/xml` ). The incorrect format will result in a "404 Not Found" error. No data requests to the gateway will ever end with a forward slash.

## Requesting Data for an Individual Device

To request data for an individual device that you have configured within the gateway, you simply need to specify the way you would like the data for the device encoded as well as the name you supplied for it. For example, the URL to request data for a device named "Breaker1" encoded as XML would be:

**http://192.168.0.1/gateway/xml/device/Breaker1**

Please take note of the format:

http://**Error! Hyperlink reference not valid.** *ADDRESS OF YOUR GATEWAY]*/gateway/*[ENCODING]*/device/*[DEVICE'S NAME]*

The red italicized text in brackets indicates the information you need to fill in depending upon your gateway's IP address, how you would like the data encoded (JSON or XML) and the name you gave to the requested device when you configured your gateway.

## Requesting a Group of Devices

To request data for a group of devices the format of the URL is like that for requesting an individual device. For example, to request data encoded as XML for a group of devices named "Breakers_West", the URL would be:

**http://192.168.0.1/gateway/xml/group/Breakers_West**

The format is:

http://**Error! Hyperlink reference not valid.** *ADDRESS OF YOUR GATEWAY]*/gateway/*[ENCODING]*/group/*[GROUP'S NAME]*

As with requesting data for an individual device, the red italicized text in brackets indicates the information you need to fill in depending upon your gateway's IP address, how you would like the data encoded (JSON or XML) and the name you gave to the group of devices when you configured your gateway.

## Filtering Requests by Point Names

When you are requesting data for a specific device or a group of devices, you can add an optional list of one or more data point names to filter the data even further. For instance, if you had a group of devices you named "Breakers_West" and you are only interested in the data points within that group named "Temperature" and "Tripped_State" encoded as XML, you can append those data point names using the following format:

**http://192.168.0.1/gateway/xml/group/Breakers_West?Temperature&Tripped_State**

Note that the string of data point names to filter by begins with a question-mark (‘?’) after the group name and each name is separated by an ampersand (‘&’). The ordering of the points’ names does not matter.

The format is the same if you are filtering within a specific device. For instance, to request the same points encoded as XML within a single Device named "Breaker1", the URL to use would be:

http://192.168.0.1/gateway/xml/device/Breaker1?Temperature&Tripped_State

---

*SPECIAL CONSIDERATIONS WHEN FILTERING BY POINT NAMES*

*Normally if you make a request for a group or device name that does not exist, the gateway returns an HTTP 404 "Not Found" error status code. When you are requesting data with a data points filtering list, you will only get an HTTP 404 status code if **none** of the points in the list are found. This is important if you are generating your request strings programmatically, as the only way to determine if a data point was not found in such a case would be to examine the encoded output for the existence of the data point names you requested in the encoded data.*

---

## Advanced (Optional XML Data Direction Annotation)

An additional option when requesting all the data from the gateway in XML format is to include the "annotateio=true" option in the request URL. This option may only be used when requesting the entire dataset from the gateway in XML. This does not apply to requesting JSON data.

**http://192.168.0.1/gateway/xml?annotateio=true**

This will include an XML attribute in the XML that indicates whether a given data point is an **input** or **output**. Note that an "input" is a *write data point* in the 460WI's configuration, and an "output" is a *read data point*. Refer the following example for reference:

```
…

<DataPointOne iotype="output">4</DataPointOne>
<DataPointTwo iotype="input">3</DataPointTwo>
…
```

*Figure 1: Example of XML segment with IO type annotation.*

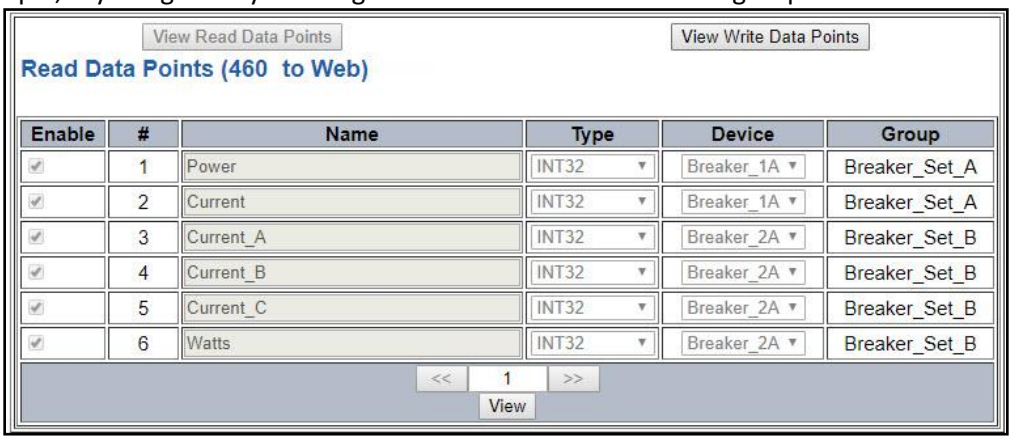
This will typically only be used for custom programmatic access to initially determine the nature of the 460WI's data such as can be seen in the included Excel example code. Please refer to the 460 Encoder XML Data Retrieval Format section for a more detailed explanation of the XML data format generated by the encoder.

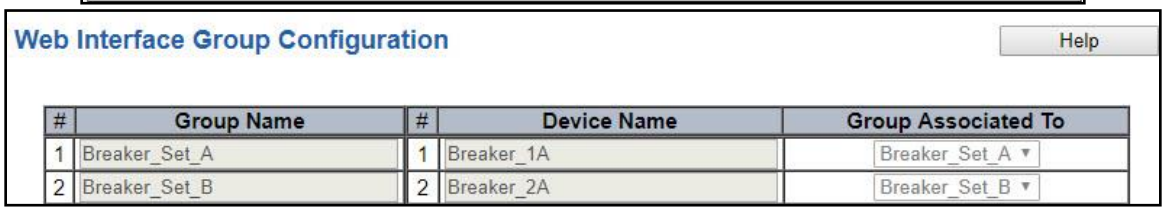
Example Output Format (GET)

This format is used when a <web server/web client> wants to read the data points configured in the gateway. We are outputting the data to <web server/web client> through a HTTP GET operation.

**NOTE**: The gateway outputs both the read data points and the write data points configured.

As an example, say the gateway is configured for two different breaker groups as shown below:




## Description of Format – Both XML and JSON

1) The <`GroupName`> and <`DeviceName`> tags contain the name of name the gevice and the group associated with that element.
2) The <`dataTimeStamp`> tag indicates the time and date when the data was read from the gateway based upon the time configured in the gateway (see Time Configuration page for more information). The format of this tag is *YYYY-MM-DD***T***HH*:*MM*:*SS*.
3) The remaining tags in the figure, for example <`Power`> and <`Current_C`>, represent user defined *Data Point* names. Their respective data values will be contained within the tags (in the above example, `237` and `36446` respectively).

## XML Output

The output to the request for http://IPADDRESS/gateway/xml will be like the following:

```
<Devices>
	<Device>
		<GroupName>Breaker_Set_A</GroupName>
		<DeviceName>Breaker_1A</DeviceName>
		<dataTimeStamp>2016-07-
	04T21:06:50</dataTimeStamp>
		<Power>237</Power>
		<Current>344</Current>
	</Device>
	<Device>
		<GroupName>Breaker_Set_A</GroupName>
		<DeviceName>Breaker_2A</DeviceName>
		<dataTimeStamp>2016-07-
	04T21:06:50</dataTimeStamp>
		<Current_A>37502</Current_A>
		<Current_B>8296</Current_B>
		<Current_C>36446</Current_C>
		<Watts>1002</Watts>
	</Device>
</Devices>
```

## JSON Output

The output to the request for http://IPADDRESS/gateway/json will be like the following:

```
[{
	"GroupName":	"Breaker_Set_A",
	"DeviceName":	"Breaker_1A",
	"dataTimeStamp":	"2016-07-04T21:06:50",
	"Power":	237,
	"Watts":	344
}, {
	"GroupName":	"Breaker_Set_A",
	"DeviceName":	"Breaker_2A",
	"dataTimeStamp":	"2016-07-04T21:06:50",
	"Current_A":	37502,
	"Current_B":	8296,
	"Current_C":	36446,
	"Watts":	1002
}]
```

## Writing Data to the Gateway

All data writes to the 460WI use the HTTP POST method. There are two primary ways through which you can post data to the 460WI. The first is using the ***application/x-www-form-urlencoded*** standard post format. The second method is to use the ***multipart-form data*** post format to post JSON or XML to the 460WI and is most easily thought of as simply uploading an XML or JSON file to the 460WI. A simple way to think of these is that the ***multipart-form data*** post method is akin to uploading an XML or JSON file to the 460WI, and the ***application/x-www-form-urlencoded*** **post** method like submitting values from a form to a web page.

## Which method to use?

Either method may be used to achieve the same results, so which method chosen depends upon several factors, not the least of which is simply whichever method is more convenient for you. A possible example where the ***multipart-form data*** post would be more convenient is in a recipe manager or similar implementation, where pre-defined XML or JSON files containing static data are written to the 460WI. It may also be the easier choice for IT backend integration where symmetry in communication models is desirable and JSON and XML parsing and generation is common practice.

The ***application/x-www-form-urlencoded*** **post** method may be more useful in situations where one or a small number of data points are frequently written to and generating XML or JSON is inconvenient.

Aside from the encoding method itself, the primary functional difference between the two methods is that the ***multipart-form data*** method may be used to write all the devices and data points configured in the 460WI with a single HTTP transaction. The ***application/x-www-form-urlencoded*** **post** method, on the other hand, can only write to a single device per HTTP transaction and thus would require a separate POST operation for each device that is to be written to.

## The multipart/form data Method

The multipart/form data post is the standard method for uploading files to web servers. This method encodes the contents of a file within the POST message body. Using this method, you will upload your data encoded in one of the supported formats (XML, JSON) and then the 460WI will decode the format, validate the contents and then perform the write operations. Note that your data need not be a "file" in the sense of a file on a computer, though of course it could be. Rather, the "file" could just as well be generated on-the-fly by an application, scripting language or library.

The exact details of the operation of the multipart/form data post method has more complexity than is suitable for discussing here. These details will generally be handled by your application, scripting language or library. There are many good resources available online describing the operations and formatting involved beginning with the current IETF specification in RFC 2388.

As a user of the 460WI, the only detail you likely need to be concerned with is the **HTTP name** attribute associated with the data. This depends on the encoding you are using. If you are using XML to encode

your data, the name attribute you should use is **`WI460XMLData`** and if you are using JSON, the name attribute you should use is **`WI460JSONData`**. Your application, scripting language or library may also

have a “filename” parameter or argument. This is ignored by the 460WI when processing an incoming POST.

Additionally, it is important to use the correct URL when posting data to the Encoder using the multipart/form method.

For XML, use the following URL: http://[Gateway’s IP]/gateway/xml

For JSON, use the following URL: http://[Gateway’s IP]/gateway/json

## The application/x-www-form-urlencoded Method

The application/x-www-form-urlencoded post method is commonly used for submitting data to websites.

The URL path format for an urlencoded post has the form:

**/gateway/device/**[DEVICE NAME]**?**[POINT 1 NAME]**=**[POINT 1 VALUE]**&**[POINT 2 NAME]**=**[POINT 2 VALUE]

Note that the URL path string begins with a question mark (?) after the device name, and more points may be added, separating each point=value pair with an ampersand (&). Also notice that unlike GET requests to read data from a device configured in the gateway, the encoding segment of the path (JSON or XML) is omitted in this case.

There is no hard limit to the number of points that can be written with a single urlencoded post, but the urlencoded string should not exceed 16,000 characters. As an extreme example, if the data points all have 32-character names and contain STRING data values that are 255 characters long, the individual POST would have to be limited to 50 data points.

Note that the urlencoded format requires the usage of the percent-encoding mechanism for data to be processed properly. In almost all cases, the actual percent-encoding will be handled transparently by your application, scripting language or library, but you should keep the possible length increase that is a side-effect or percent-encoding in mind. It is recommended to leave some headroom because certain characters, for example the quotation mark (“) character will be encoded as “%22” and therefore consume three characters rather than one. In the case of the 460WI, this is generally only a concern for STRING data point types. Refer to RFC 3986 for a detailed explanation of percent-encoding.

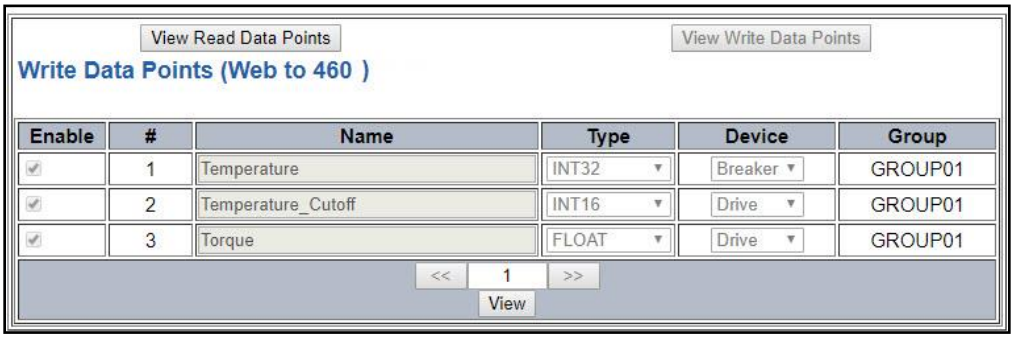
Example Input Format (POST)

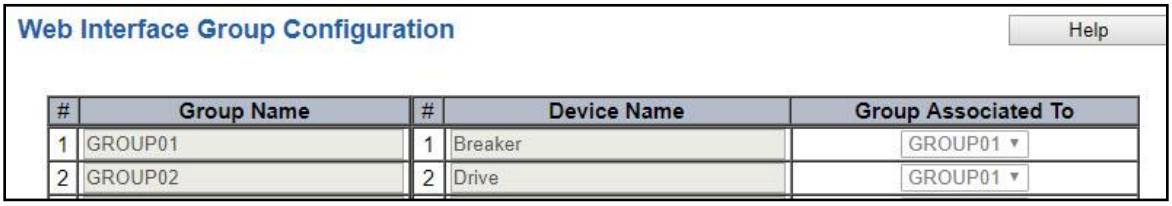
This format is used when an external web client writes XML/JSON data to the gateway through a HTTP POST operation. The input format is the same as the output format except that the GroupName and dataTimeStamp may be omitted. If they included, they are ignored.

**Important Notes:**

- Only data points configured as write data points may be written to.
- The value for STRING datatypes must between double quotation marks.

A gateway is configured for a breaker group and a drive group as shown below:




## Minimum XML Input

At minimum, the XML POST is the `<Devices>` root element, and at least one `<Device>` child-element containing the matching `<DeviceName>` tag of the device to be written to, and finally, at least one writable data point tag/value pair.

The following example represents a minimum XML POST. This POST will write a value 100 to the point named “Temperature” which belongs to the device named “Breaker”:

```
<Devices>
  <Device>
    <DeviceName>Breaker</DeviceName>
    <Temperature>100</Temperature>
  </Device>
</Devices>
```

## Expanded XML Input

The following example is slightly more complicated and writes one data point to one device and two data points to a different device. Expanding on this pattern, it is possible to write to every writable data point of every device configured in the gateway with a single XML write.

```
<Devices>
  <Device>
    <DeviceName>Breaker</DeviceName>
    <Temperature>100</Temperature>
  </Device>
  <Device>
    <DeviceName>Drive</DeviceName>
    <Temperature_Cutoff>32</Temperature_Cutoff>
    <Torque>15.2</Torque>
   </Device>
</Devices>
```

## Special Note for XML Strings

There is an additional consideration when posting data points with the STRING data type. "Unsafe" XML characters, such as "<" and "&" that might appear in STRING data should be converted to their respective XML entity-references. In general, your application (ex. scripting language or library), will either handle this automatically or provide easy-to-use mechanisms to accomplish this for you. When the gateway parses the XML data written to it, it will convert these entity-references back to their ASCII equivalent characters before writing the STRING data to the other protocol. The reverse is also true when reading STRING data from the other protocol.

## Minimum JSON Input

At minimum, a valid JSON post must include the root array, one Device Object, which must include the `DeviceName` name/value pair member to identify the device followed by at least one writeable name/value pair member.

The following example represents a minimal JSON POST. This POST will write a value 100 to the point named "Temperature" which belongs to the device named "Breaker":

```
[
  {
    "DeviceName": "Breaker",
    "Temperature": 100
  }
]
```

## Expanded JSON Input

The following example is slightly more complicated and writes one data point to one device and two data points to a different device. Expanding on this pattern, it is possible to write to every writable data point of every device configured in the gateway with a single JSON write.

```
[
  {
    "DeviceName": "Breaker",
    "Temperature": 100
  },
  {
    "DeviceName": "Drive",
    "Temperature_Cutoff": 32,
    "Torque": 15.2
  }
]
```

## Special Note for JSON Strings

There is an additional consideration when writing STRING data types. The gateway recognizes and will convert the standard JSON escape sequences to their ASCII equivalents when passing the STRING value to the mating protocol. The reverse is also true when reading STRING data from the mating protocol. The exception to this rule is the “\u” Unicode code-point escape sequence, which will be passed as-is.

## Mapping - Transferring Data Between Devices

There are 5 ways to move data from one protocol to the other. You can combine any of the following options to customize your gateway as needed.

**Option 1 – Data Auto-Configure Mappings:** The gateway will automatically take the data type (excluding strings) from one protocol and look for the same data type defined in the other protocol. If there isn't a matching data type, the gateway will map the data to the largest available data type. See Data Auto-Configure section for more details.

**Option 2 – String Auto-Configure:** The gateway will automatically take the string data type from one protocol and map it into the other. See String Auto-Configure section for more details.

**Option 3 – Manual Configure Mappings:** If you don't want to use the Auto-Configure Mappings function, you must use the manual mapping feature to configure translations.

**Option 4 – Manipulation/Scaling:** You can customize your data by using math operations, scaling, or bit manipulation. See Data Mapping-Explanation section for more details.

**Option 5 – Move Diagnostic Information:** You can manually move diagnostic information from the gateway to either protocol. Diagnostic information is not mapped in Auto-Configure Mappings Mode. See Diagnostic Info section for more details.

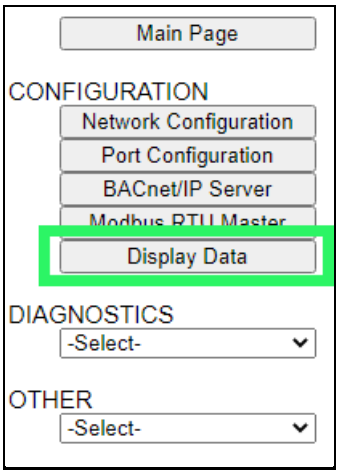
**Going from Manual Mapping to Auto-Mapping will delete ALL mappings and manipulations configured.**

## Display Mapping and Values

The Display Data and Display String pages are where you can view the actual data for each mapping that is set up.

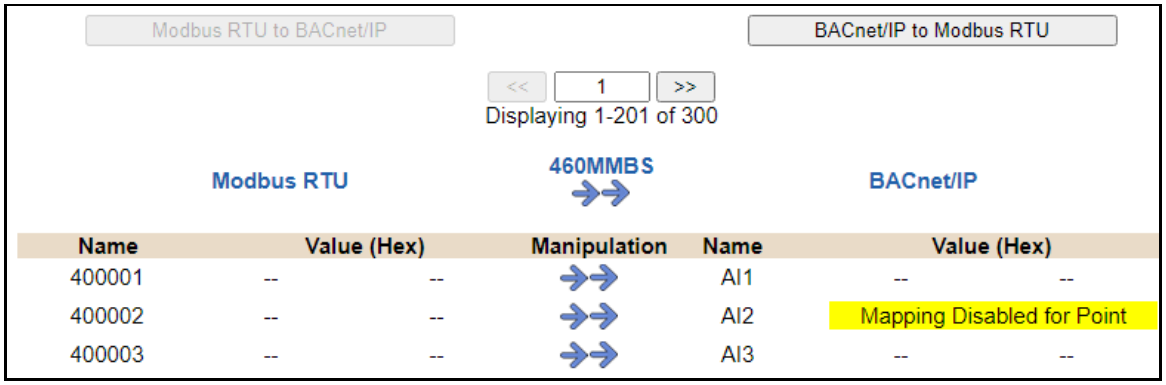
## Display Data

Click the **Display Data** button to view how the data is mapped and what the values of each mapping are.




Here you will see how each data point (excluding strings) is mapped. To view, select the device from the dropdown menu and click **View** to generate the information regarding that device. Then select either the ***Protocol 1 to Protocol 2*** or ***Protocol 2 to Protocol 1*** button, correlating to the direction you wish to see the data.

This page is very useful when verifying that all data is mapped somehow from one protocol to another. If a data point is not mapped, it will display on this page in a yellow highlighted box. The Display Data page will display up to 200 mappings per page, simply navigate to the next page for the additional mapping to display.




In the above example, we see the following:

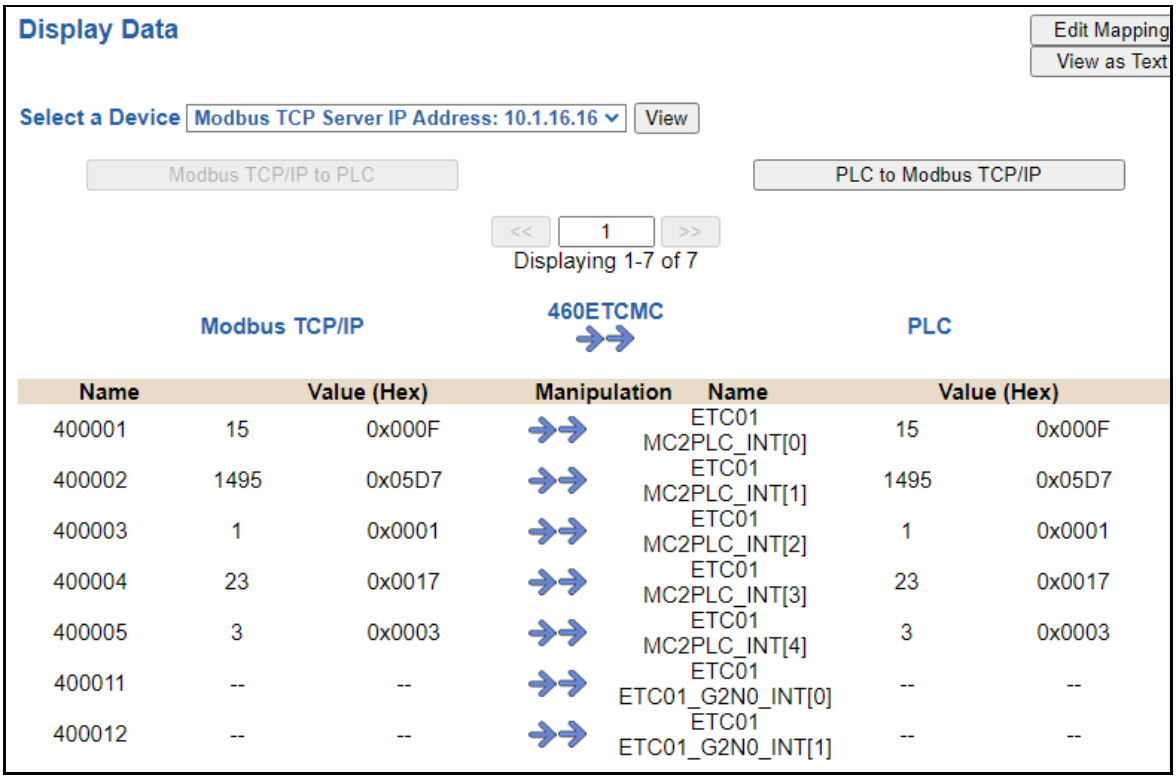
- Modbus register 400001 from Slave 1 is being mapped to AI1 on BACnet
- Nothing is being moved from Modbus register 400002 to AI2 on BACnet because the mapping is disabled
- Modbus register 400003 from Slave 1 is being mapped to AI3 on BACnet

**NOTE**: If a data point is mapped twice, only the first instance of it will show here. EX: If Modbus 400001 & 400040 from Slave 1 are both mapped to AI1, only 400001 will show as being mapped to AI1.

If there are values of "- - "on this page, it indicates that the source has not yet been validated and no data is being sent to the destination.

The example below reflects the Modbus to PLC flow of data. The Modbus (left side) is the source and the PLC (right side) is the destination.

- The 460 gateway has received valid responses from Modbus registers 400001- 400005 and therefore can pass the data on to the PLC tag called MC2PLC_INT.
- The 460 gateway has NOT received valid responses from Modbus register 400011 & 400012. As a result, the data cannot be passed to the PLC tag ETC01_GN0_INT2 and indicates so by using "- - "in the value column of the table.

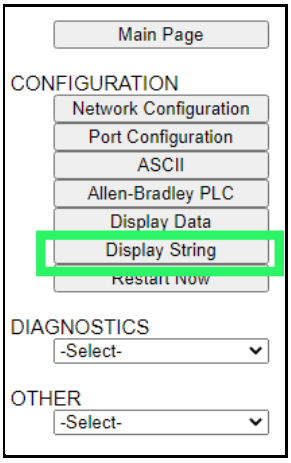



To view the actual data mappings, click the **Edit Mapping** button. For more details, see the Data Mapping-Explanation section.

To view the data mappings purely as text, click the **View as Text** button. For more details, see the View Data Mapping as Text section.

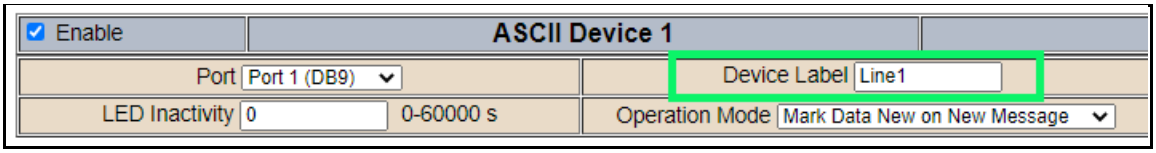
## Display String

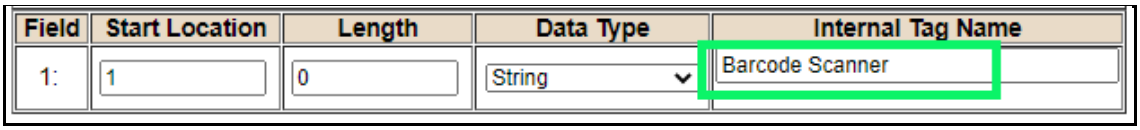
Click the **Display String** button to view what the values of each Parsing and/or Concatenating strings are, you can also click on the Edit Mapping to view the mapping of each string.




To view the source or destination groups from a string, click the dropdown menu to generate the information regarding that device. The string data will be displayed in both Hex and ASCII (only the ASCII data is sent). The example below shows data that is coming from the source device. A group will be displayed for each Parsing/Concatenating String field that is configured.

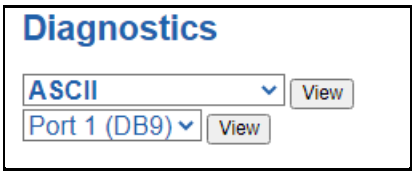
In the Group drop down, “Line1” is defined on the ASCII Device configuration page and “Barcode Scanner” is defined in the ASCII Parsing configuration.



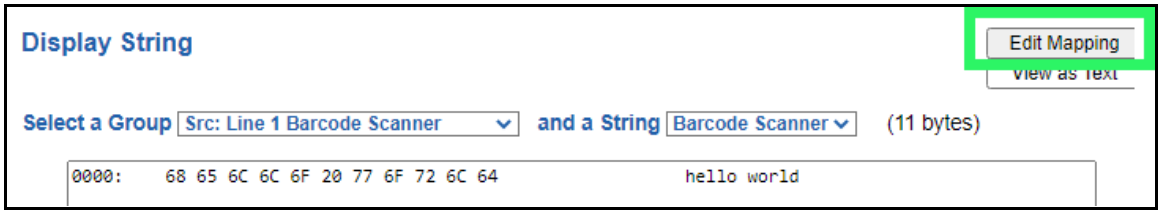



If there are values of "Data Not Valid "on this page, it indicates that the source has not been validated yet and no data is being sent to the destination.

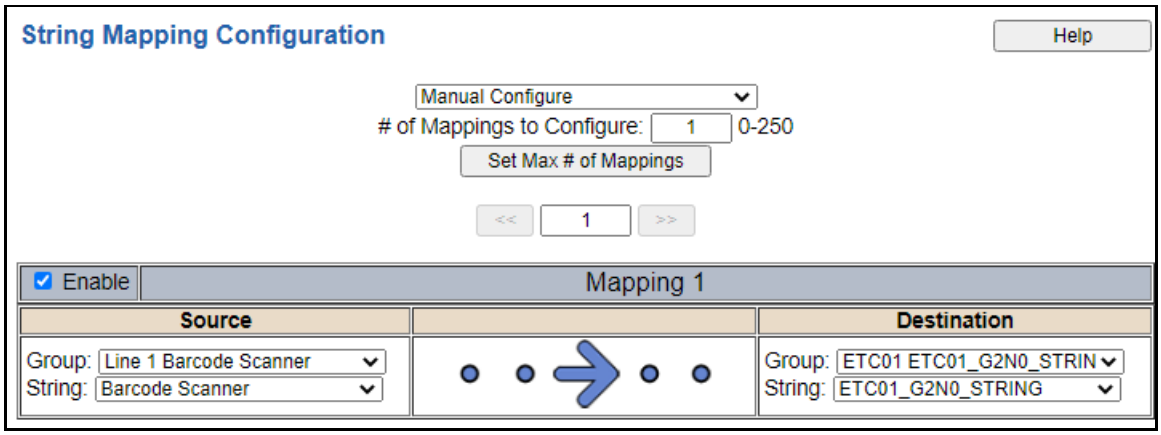
**NOTE:** You can view the whole string data by clicking on **Diagnostics Info** drop down and navigating to ASCII Diagnostics page. You will also have to select the port you want to view in the dropdown below ASCII.




To view the string mappings, click the **Edit Mapping** button. For more details see the **String Mapping-Explanation** section.




**NOTE: Only String data types can be mapped to another String data type.**




To view the string mappings purely as text, click the **View as Text** button. For more details see the **View String Mapping** as Text section.

## Display String use case

Sending a message of “RTA,Support,Rocks” from an ASCII device to the RTA unit. The ASCII Parsing Configuration would look like my example below. There are more detailed examples of what all the fields represent in the ASCII Parsing section.

**ASCII Device 1** (Line1)

Max Number of Fields: 3 1-50 | Min Number of Fields: 1 1-50

Parsing Delimiter: , 44 0x2c

Update Fields

| Field | Start Location | Length | Data Type | Internal Tag Name |
|---|---|---|---|---|
| 1: | 1 | 0 | String | Header 1 |
| 2: | 1 | 0 | String | Header 2 |
| 3: | 1 | 0 | String | Header 3 |

The message is broken up into 3 “Groups” or Parsing fields.

**Display String** — Edit Mapping / View as Text

Select a Group: Src: Line1 Header 1 and a String: Header 1 (3 bytes)

```
0000:    52 54 41                                          RTA
```

**Display String** — Edit Mapping / View as Text

Select a Group: Src: Line1 Header 2 and a String: Header 2 (7 bytes)

```
0000:    53 75 70 70 6F 72 74                              Support
```

**Display String** — Edit Mapping / View as Text

Select a Group: Src: Line1 Header 3 and a String: Header 3 (5 bytes)

```
0000:    52 6F 63 6B 73                                    Rocks
```

To view the Entire message, click on the Diagnostic drop down, select Diagnostics Info. Select ASCII, click view, select your Port. Whole data will be in the Last Message Sent Diagnostic box.

**Diagnostics**

ASCII View

Port 1 (DB9) View

Last Message Sent (17 bytes)

```
0000:    52 54 41 2C 53 75 70 70 6F 72 74 2C 52 6F 63 6B   RTA,Support,Rock
0016:    73                                                s
```

## Data and String Mapping – Auto-Configure

The Auto-Configure function looks at both protocols and will map the data between the two protocols as best as it can so that all data is mapped. Inputs of like data types will map to outputs of the other protocols like data types first. If a matching data type cannot be found, then the largest available data type will be used. Only when there is no other option is data truncated and mapped into a smaller data type.

If the Auto-Configure function does not map the data as you want or you want to add/modify the mappings, you may do so by going into Manual Configure mode.

The following are examples of the Auto-Configure function.

1) This example shows a common valid setup.

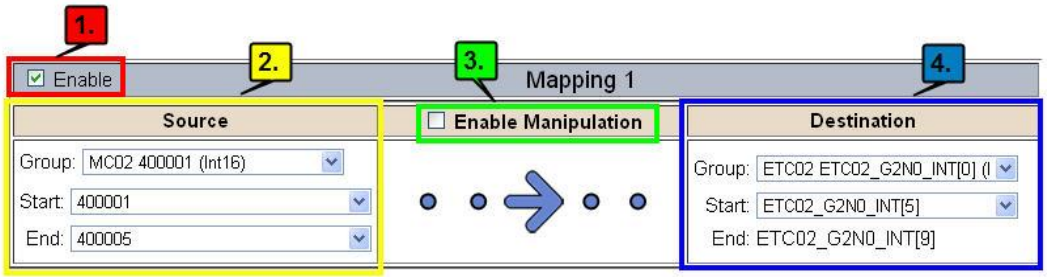
   a. Both Source values were able to be mapped to a corresponding Destination value.

2) This example shows how Auto-Configure will make its best guess.

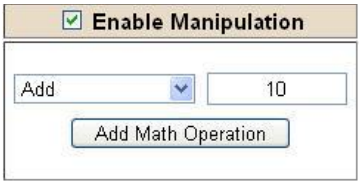
   a. The 32-bit Float from the Source location could not find a matching Destination data-type. After all other like data types were mapped, the only data type available was the 2nd 32-bit Uint data type. Auto-Configure was completed even though the data in the Float will be truncated.

## Data Mapping – Explanation

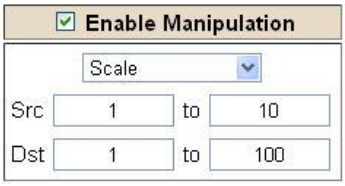
Below are the different parts that can be modified to make up a data mapping.



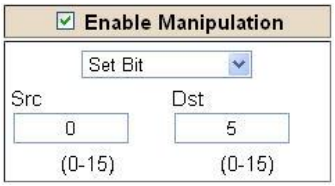

1) Enable (red box above): Check to enable mapping. If not checked, this mapping is skipped.
2) Source Field (yellow box above):
   a) Group - Select the data group you set up in the protocol config to use for this mapping.
   b) Start - This is the starting point for this mapping.
   c) End - This is the final point to be included for this mapping.
3) Manipulation Area (green box above):
   a) Enable the Data Manipulation. This can be enabled for any mapping.
   b) Click **Add Math Operation** for each operation needed. Up to 3 are allowed unless you are using the Scale, Set Bit, or Invert Bit functions. If using Scale, Set Bit, or Invert Bit, then only 1 operation is allowed.
   c) Select the Operation(s) to perform.
      i) Math Operations are performed in the order they are selected.
      ii) If more than one point is selected on the source, the Math Operations will be performed on every point.
   d) Enter the value(s) for the operation.




*Example of Add (similar for Subtract, Multiple, Divide, and MOD). This will add a value of 10 to the source field before it is written to the destination field.*




*Example of Scale. This will scale the source values from 1-10 into 1-100 for the destination.*




*Example of Set Bit (similar to Invert Bit). This will take the value of the 0th source bit and copy it into the value of the 5th destination bit.*

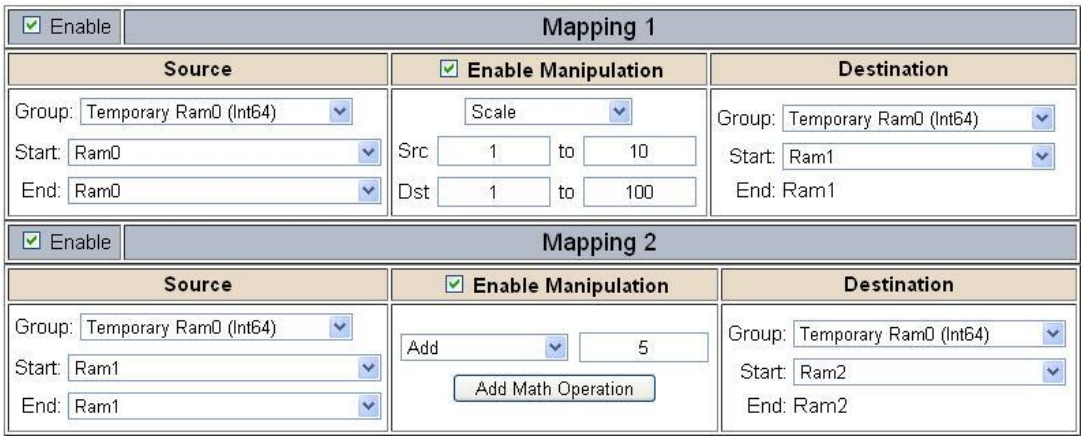
4) Destination Field (blue box above):
   a) Group - Select the data group you set up in the protocol config to use for this mapping.
   b) Start - This is the starting point for where the data is being stored.
   c) End - The End point is derived from the length of the source and cannot be modified.

## Data Mapping – Adding Diagnostic Information

Data Mapping offers 5 different types of information in addition to any scan lines specified for each protocol.

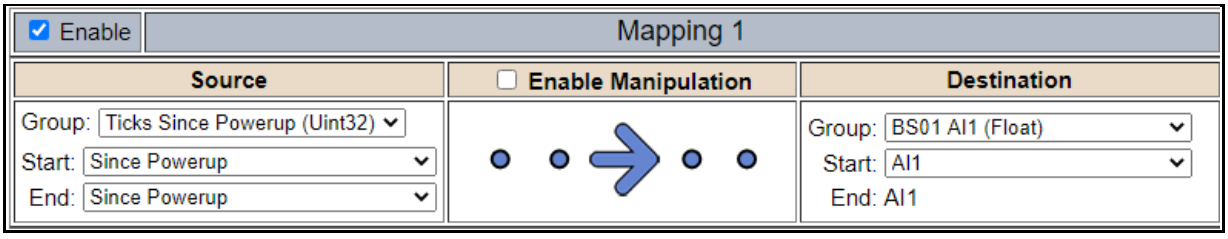
**IMPORTANT NOTE:** Only add Diagnostic Information **AFTER** both sides of the gateway have been configured. If changes to either protocol are made after diagnostic information has been added to the mapping table, it is necessary to verify all mappings. Remapping may be necessary.

1) **Temporary Ram (Int64)**
   a) This offers five levels of 64bit Integer space to assist in multiple stages of math operations. For example, you may wish to scale and then add 5. You can set up a single translation to scale with the destination as the temporary ram. Then another translation to add 5 with the source as the temporary ram.
   b) The gateway will automatically convert the Source to fit the Destination, so there is no need for Int 8, 16, 32 since the 64 may be used for any case.



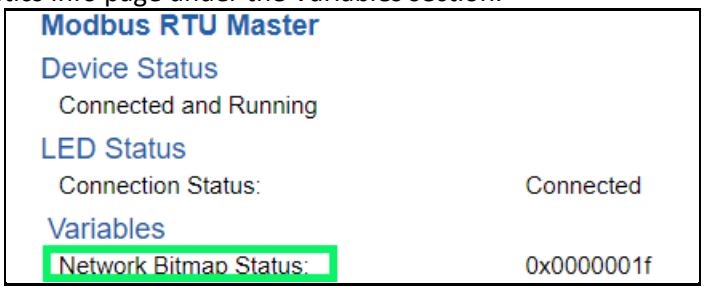

*In this example, Ram0 is scaled into Ram1. Ram1 is then increased by 5 and stored into Ram2. Ram0 and Ram2 could be considered a source or destination group.*

2) **Temporary Ram (Double)**
   a) This is like the Temporary Ram (Int 64), except manipulations will be conducted against the 64bit floating point to allow for large data.
3) **Ticks Per Second**
   a) The gateway operates at 200 ticks per second. This equates to one tick every 5ms. Thus, mapping this to a destination will give easy confirmation of data flow without involving one of the two protocols. If data stops on the destination end, then the RTA is offline.



**4) XY_NetBmpStat**

a) If a protocol is a Client/Master, there is a Network Bitmap Status that is provided on the Diagnostics Info page under the Variables section.



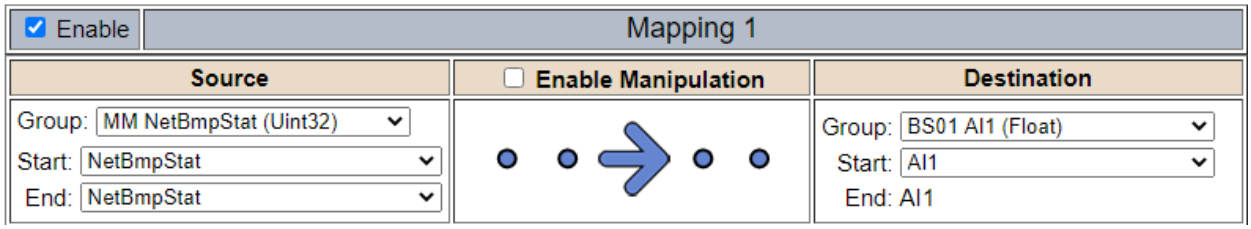

b) Since a Client/Master may be trying to communicate with multiple devices on the network, it may be beneficial to know if a Server/Slave device is down. By using this Network Bitmap Status, you can expose the connection statuses of individual devices. **Values shown are in HEX.**

i) 0x00000002 shows that only device 2 is connected
ii) 0x00000003 shows that only devices 1 and 2 are connected
iii) 0x0000001f shows that all 5 devices are connected (shown in image above)

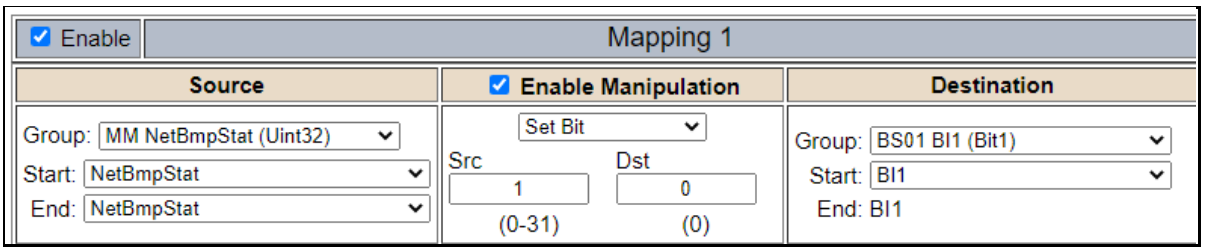
c) There are multiple ways to map the NetBmpStat.

**Option 1:** Map the whole 32bit value to a destination. Example below shows the NetBmpStat is going to an Analog BACnet object. Using a connection of 5 Modbus Slave devices AI1 will show a value of 31.0000. Open a calculator with programmer mode and type in 31, this will represent bits 0 – 4 are on. This mean all 5 devices are connected and running.

If using an AB PLC with a Tag defined as a Dint, then expand the tag within your RSlogix software to expose the bit level and define each bit as a description such as device1, device2, etc.




**Option 2:** You can extract individual bits from the NetBmpStat by using the Set Bit Manipulation and map those to a destination. You'll need a mapping for each device you want to monitor. Example below shows Modbus device 2 (out of 5) is being monitor to a BACnet Binary Object. You can define the object in the BACnet Name configuration.



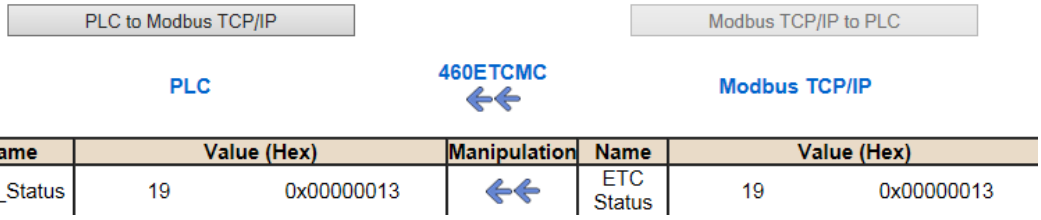
**5) Status_XY**

a) There are two Statuses provided, one for each protocol. This gives access to the overall status of that Protocol. Each Bit has its own meaning as follows:

***Common Status: 0x000000FF (bit 0-7) 1st byte***

| Hex: | Bit Position: | Decimal: | Explanation: |
|---|---|---|---|
| 0x00 | 0 | 0 | if we are a Slave/Server |
| 0x01 | 0 | 1 | if we are a Master/Client |
| 0x02 | 1 | 2 | connected (0 not connected) |
| 0x04 | 2 | 4 | first time scan |
| 0x08 | 3 | 8 | idle (usually added to connected) |
| 0x10 | 4 | 16 | running (usually added to connected) |
| 0x20 | 5 | 32 | bit not used |
| 0x40 | 6 | 64 | recoverable fault |
| 0x80 | 7 | 128 | nonrecoverable fault |

For this example, the ETC Status is mapped to a PLC tag called PLC_Status




| Name | Value (Hex) | | Manipulation | Name | Value (Hex) | |
|---|---|---|---|---|---|---|
| PLC_Status | 19 | 0x00000013 | ← | ETC Status | 19 | 0x00000013 |

**Example:** ETC Status is 0x00000013 (19 decimal), here is the break down

| Hex | Bit | Decimal | Explanation |
|---|---|---|---|
| 0x01 | 0(on) | 1 | if we are a Master/Client |
| 0x02 | 1(on) | 2 | connected (0 not connected) |
| 0x10 | 4(on) | 16 | running (usually added to connected) |
| Total: | 0x13 | 19 | |

***External Faults: 0x0000FF00 (bit 8-15) 2nd byte***

| Hex: | Bit Position: | Decimal: | Explanation: |
|---|---|---|---|
| 0x00 | 8 | 0 | local control |
| 0x01 | 8 | 256 | remotely idle |
| 0x02 | 9 | 512 | remotely faulted |
| 0x04 | 10 | 1,024 | idle due to dependency |
| 0x08 | 11 | 2,048 | faulted due to dependency |

***Recoverable Faults: 0x00FF0000 (bit 16-23) 3rd byte***

| Hex: | Bit Position: | Decimal: | Explanation: |
|---|---|---|---|
| 0x01 | 16 | 65,536 | recoverable fault - timed out |
| 0x02 | 17 | 131,072 | recoverable fault - Slave err |

***Non-Recoverable Faults  0xFF000000 (bit 24-31) 4th byte***

| Hex: | Bit Position: | Decimal: | Explanation: |
|---|---|---|---|
| 0x01 | 24 | 16,777,216 | nonrecoverable fault – task fatal err |
| 0x02 | 25 | 33,554,432 | nonrecoverable fault – config missing |
| 0x04 | 26 | 67,108,864 | nonrecoverable fault – bad hardware port |
| 0x08 | 27 | 134,217,728 | nonrecoverable fault – config err |
| 0x10 | 28 | 268,435,456 | Configuration Mode |
| 0x20 | 29 | 536,870,912 | No Ethernet Cable Plugged In |

For this example, the MC Status is mapped to a PLC tag called MC_Status

PLC to Modbus TCP/IP | Modbus TCP/IP to PLC

PLC — 460ETCMC — Modbus TCP/IP

| Name | Value (Hex) | | Manipulation | Name | Value (Hex) | |
|---|---|---|---|---|---|---|
| MC_Status | 65601 | 0x00010041 | ← | MC Status | 65601 | 0x00010041 |

**Example:** MC Status is 0x00010041 (65601 decimal), here is the break down, we know that bytes 1 and 3 are being used, so here is the break down,

***Common Status:***

| Hex: | Bit: | Decimal: | Explanation: |
|---|---|---|---|
| 0x01 | 0(on) | 1 | if we are a Master/Client |
| 0x40 | 6(on) | 64 | recoverable fault |

***Recoverable Faults:***

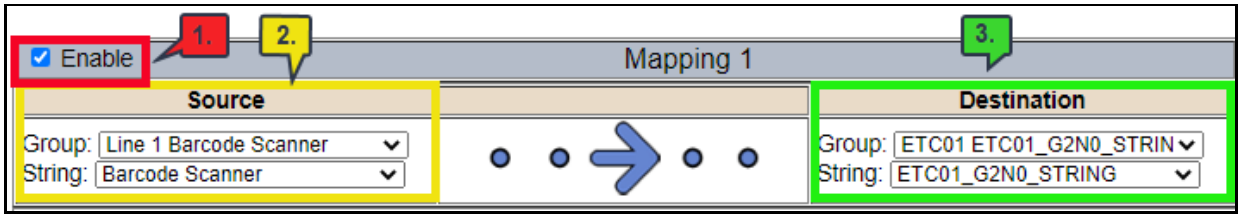
| Hex: | Bit: | Decimal: | Explanation: |
|---|---|---|---|
| 0x01 | 16 | 65,536 | recoverable fault – timed |

| Total: | 0x010041 | 65,601 |
|---|---|---|

## String Mapping – Explanation

Below are the different parts that can be modified to make up a string mapping.

String data types can only be mapped to other string data types. There is no manipulation that can be done on the string.

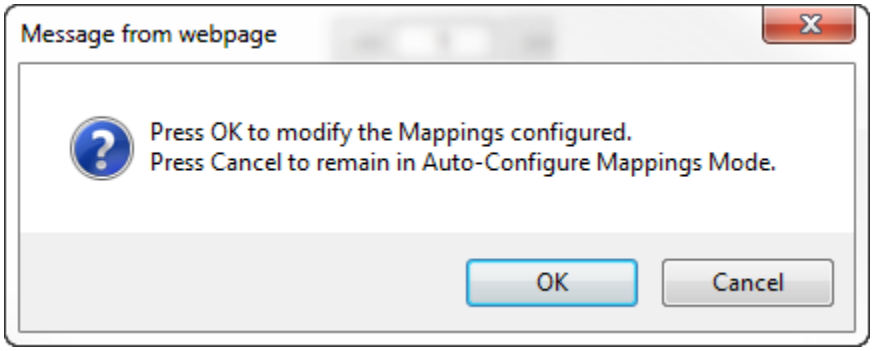



1) Enable (red box above): Check to enable mapping. If not checked, this mapping is skipped.
2) Source Field (yellow box above):
   a) Group - Select the string data group you set up in the protocol config to use for this mapping.
   b) String - This is the string used for this mapping.
3) Destination Field (green box above):
   a) Group - Select the string data group you set up in the protocol config to use for this mapping.
   b) String - This is the string where the data is being stored.

## Mapping – Auto-Configure Mode to Manual Configure Mode

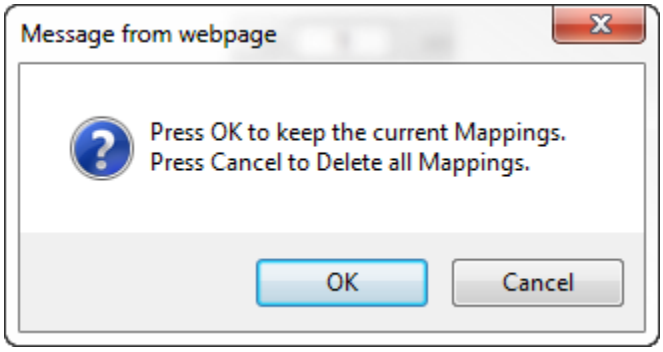
To transition from Auto-Configure Mapping Mode to Manual Configure Mode, click the dropdown at the top of the Mapping Configuration page and select Manual Configure.

After you click this button, you will be prompted to confirm if this is really what you want to do.




Click **OK** to proceed to Manual Configure Mode or click **Cancel** to remain in Auto-Configure Mappings Mode.

Once OK is clicked, there are 2 options on how to proceed from here.




1) To keep the mappings that are already configured press **OK**.

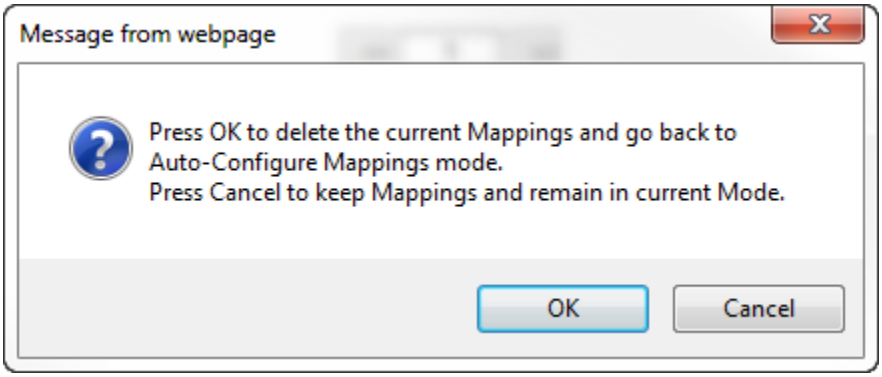
    a) You would want this option if you are adding additional mappings or you want to modify the mapping(s) that already exist.

2) To delete the mappings that are already there and start over press **Cancel**.

To modify the number of mappings, enter a number in the text field next to **# of Mappings to Configure** and click the **Set Max # of Mappings** button. You can always add more mappings if needed.

## Mapping – Manual Configure Mode to Auto-Configure Mode

To transition from Manual Configure Mode to Auto-Configure Mapping Mode, click the dropdown menu at the top of the Mapping Configuration page and select Auto-Configure Mappings.




Click **OK** to proceed to delete all current mappings and go back to Auto-Configure Mappings Mode. Click **Cancel** to keep all mappings and remain in Manual Configure Mode.

**NOTE**: Once you revert to Auto-Configure Mapping Mode there is no way to recover the mappings you lost. Any mappings you previously have added will be deleted as well.

# View as Text

## Data Mapping

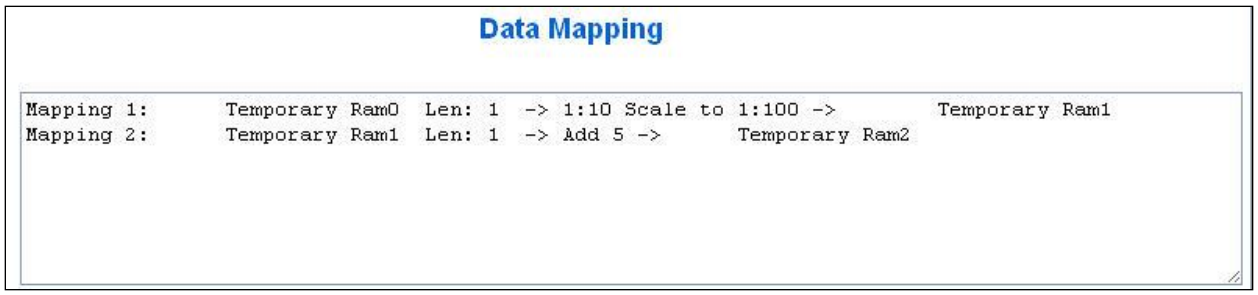
The View as Text page displays the point to point mapping(s) you set up in the Data Mapping section. This will also display any manipulation(s) that are configured.

Each line on this page will read as follows:

> **Mapping** *number*: *source point* **Len:** *Number of points mapped* **->** *manipulation (if blank then no manipulation)* **->** *destination point*

If you are looking for a specific point to see if it is mapped, you can do a find in this text box for your point in question. Example: you defined 20 Registers starting at register 1 and want to see if 400011 is mapped. If it is not in this text box, then it is not mapped, and no data will be transferred.

This is the text display for the example shown under the *Data Mapping- Adding Diagnostic Information* section.



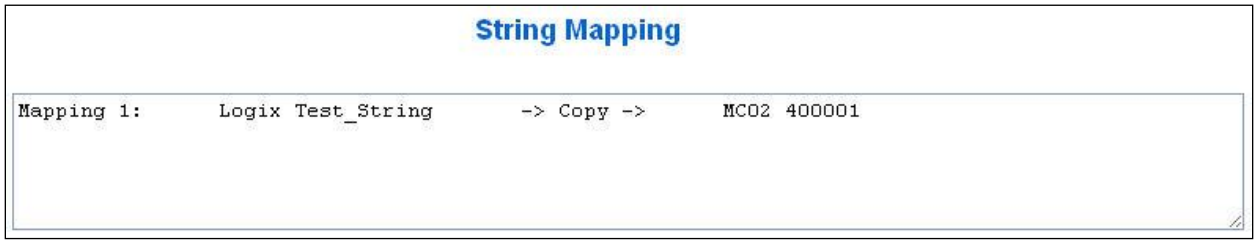

## String Mapping

The View as Text page displays the string mapping(s) you set up in the String Mapping section.

Each line on this page will read as follows:

> **Mapping** *number*: *source point* **-> Copy ->** *destination point*

If you are looking for a specific point to see if it is mapped, you can do a find in this text box for your point in question. Example: you defined 20 String Tags in the PLC and want to see if "Test_String" in the Logix PLC is mapped. If it is not in this text box, then it is not mapped, and no data will be transferred.



## Base Triggering – Data Validiation Triggering

With Base Triggering, you will be marking data as “Invalid” and force RTA Master/Controller/Client protocols to read all the read data points sources until ALL source protocols data is valid. You will be able to utilize the Handshake to map over to Technology Trigger and/or back over to your source protocol for reference.

## How does this work?

1) Map the Triggering Variable (Source) over to Trigger **#** (Dest).
2) If Trigger **#** value changes states mark all Trigger **#** protocols read data as “Invalid”.
3) Read all source read data points until ALL source read data is valid.
4) Handshake **#** value is set equal to Trigger **#** value.
5) Map Handshake **#** to reference data point.
   **Note**: **#** is an internal reference to the Server/Slave number you are settings up. **ex**. RTA Server/Slave products can only be Trigger 1 and Handshake 1 since we are only 1 device. If RTA is a Master/Client, then you can have a Trigger# for each server/slave connected too.

## How do you set this up?

In this example I’m using a 460MCBS. My Building Automation System wants to verify that all data read from Modbus TCP/IP Server is valid.

1) Add an extra Analog Output for your Trigger. This tells the RTA to mark all data invalid.

**Write Data Groups** (BACnet/IP to 460MCBS)

| Data Group | Object Type | Starting Object | # of Objects |
|---|---|---|---|
| 1 | Analog Output (32 Bit Float) | 1 | 21 |
| 2 | Binary Output | 1 | 0 |
| 3 | CharacterString Value | 51 | 0 |

   a) You can define AI21 as your validation name in the Setup BACnet Names Configuration.

Setup BACnet Names, Units, and COV

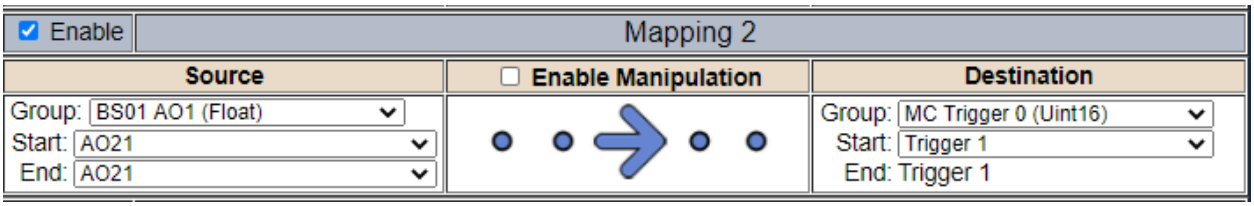
| | | | | | |
|---|---|---|---|---|---|
| 21 | G01 | Data Validation Trigger | Other | no-units | 1.000000 |

2) Add another Analog Input as reference for when data has been validated. When you write from AO21 to validate data, the RTA will reply to AI40 saying “validation complete”.

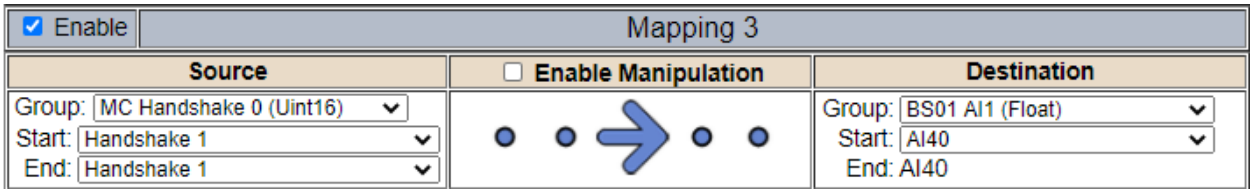
| Data Group | Object Type | Starting Object | # of Objects |
|---|---|---|---|
| 1 | Analog Input (32 Bit Float) | 1 | 40 |
| 2 | Binary Input | 1 | 0 |
| 3 | CharacterString Value | 1 | 0 |

| | | | | | |
|---|---|---|---|---|---|
| 40 | G01 | Data Validation Result | Other | no-units | 1.000000 |

3) Within the Data Mapping page manually add 2 additional mappings.
4) The first mapping is going to be the Data Validation Triggering. AO21 will write to the RTA, MC Trigger 1 will mark data invalid.




5) The second mapping, the MC Handshake will increment that all data is validated and write to AI21 “all data is validated”. The value of AI40 and AO21 should be the same.



## Security Configuration

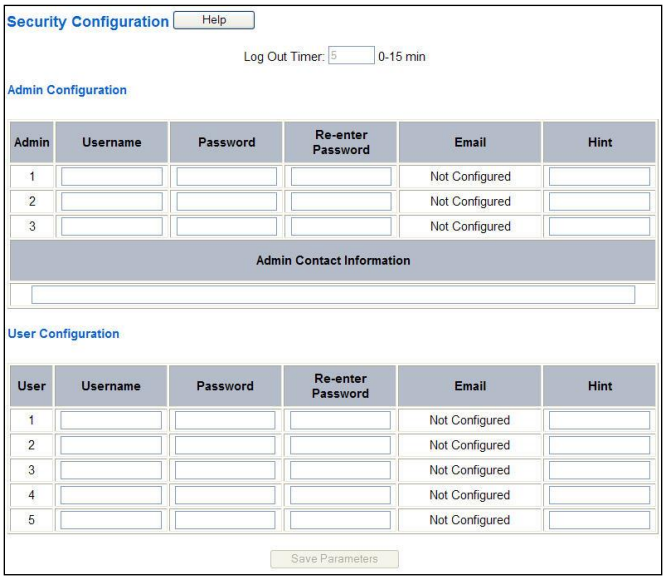
To setup security on the 460 gateway, navigate to **Other->Security Configuration**. You can configure Security for 3 administrators, 5 users, and 1 guest.

### THIS IS NOT A TOTAL SECURITY FEATURE

The security feature offers a way to password protect access to diagnostics and configuration on the network. The security feature does not protect against "Air Gap" threats. If the gateway can be physically accessed, security can be reset. All security can be disabled if physical contact can be made. From the login page, click the Reset Password button twice. You will be forced to do a hard reboot (power down) on the gateway within 15 minutes of clicking the button. This process should be used in the event a password is forgotten.

**Note:** Only Admins have configuration access to all web pages.

1) Log Out Timer: The system will automatically log inactive users off after this period of time. **NOTE**: A time of 0 means that the user will not be automatically logged off. Instead, they must manually click the **Logout** button.
2) Username: Enter a username, max of 32 characters.
3) Password: Enter a password for the username, max of 32 characters, case sensitive.
    a. Re-enter the Password
4) E-mail: In case the password was forgotten, a user can have their password e-mailed to them if e-mail was configured.
5) Hint: A helpful reminder of what the password is.

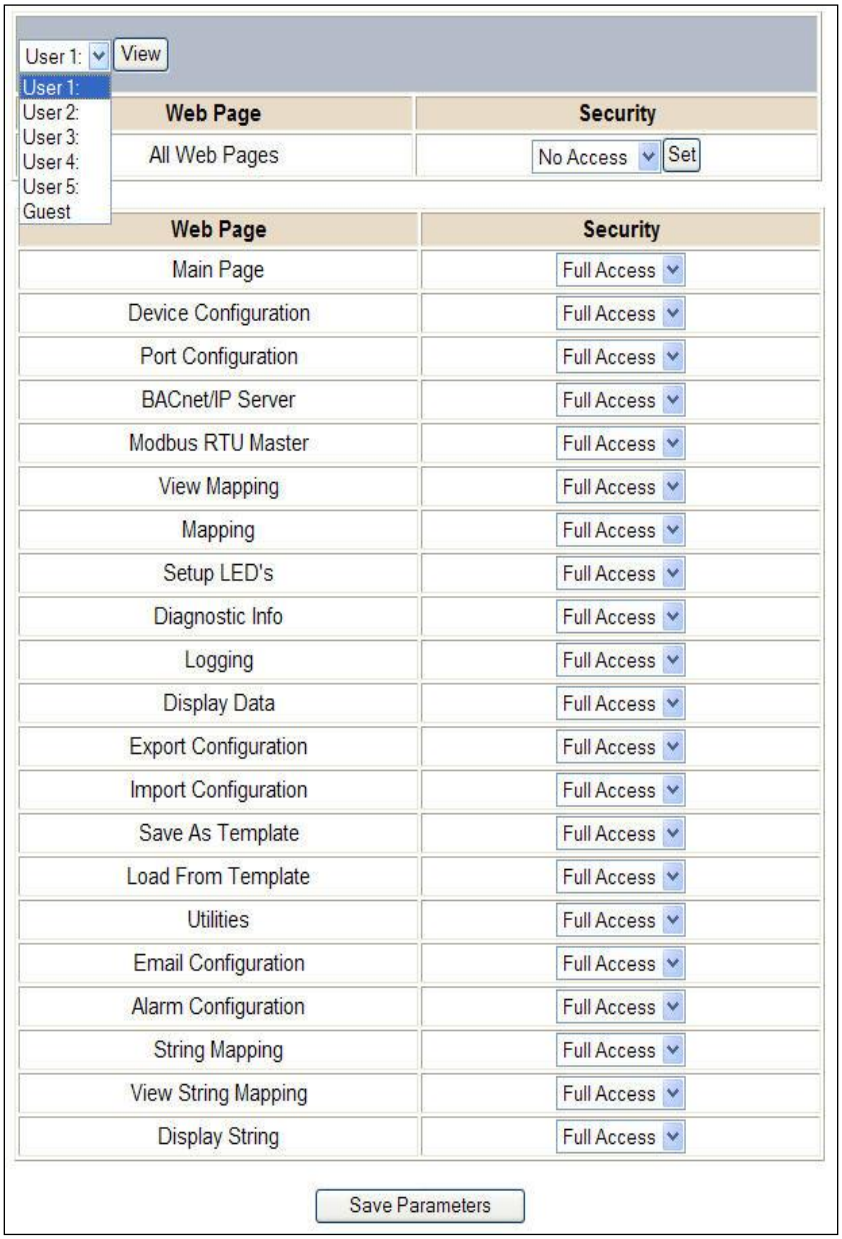


## Security Configuration-Security Levels

Each webpage in the gateway can have a separate security level associated with it for each user.

Security Levels:

1) **Full Access**: Capability to view and configure a web page.
2) **View Access**: Capability to view a web page, but cannot configure parameters.
3) **No Access**: No capability of viewing the web page and page will be removed from Navigation.



## Security - Log In

**Username**: Name of the user to login.

**Password**: Password of the user to login.

**Log In:** If login is successful, the user will be redirected to the Main Page.

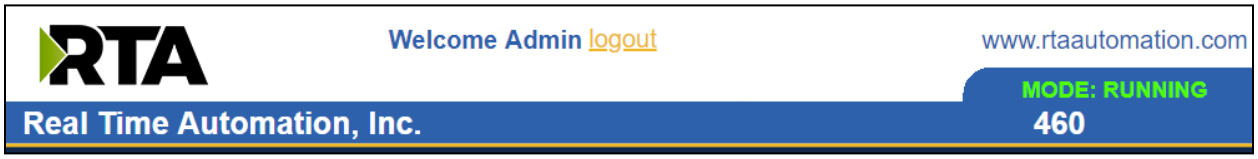
**Send Password to Email:** Sends the specified User's Password to the email configured for that user.

**Display Hint:** Displays the hint specified for the User if one was set up.

**Reset Password:** This is used to reset security settings. Confirm reset password must be selected to confirm this action. Once confirmed, there is a 15 minute window to do a hard reset of the gateway by physically removing and restoring power from the gateway. Once power is restored, you may navigate to the IP address of the gateway as normal.

## Security - Log Out

Once a user is done with a session they may click **logout** at the top of any page. The user may also be logged out for inactivity based off of the Log Out Timer specified during the configuration.




**Closing the browser is not sufficient to log out.**

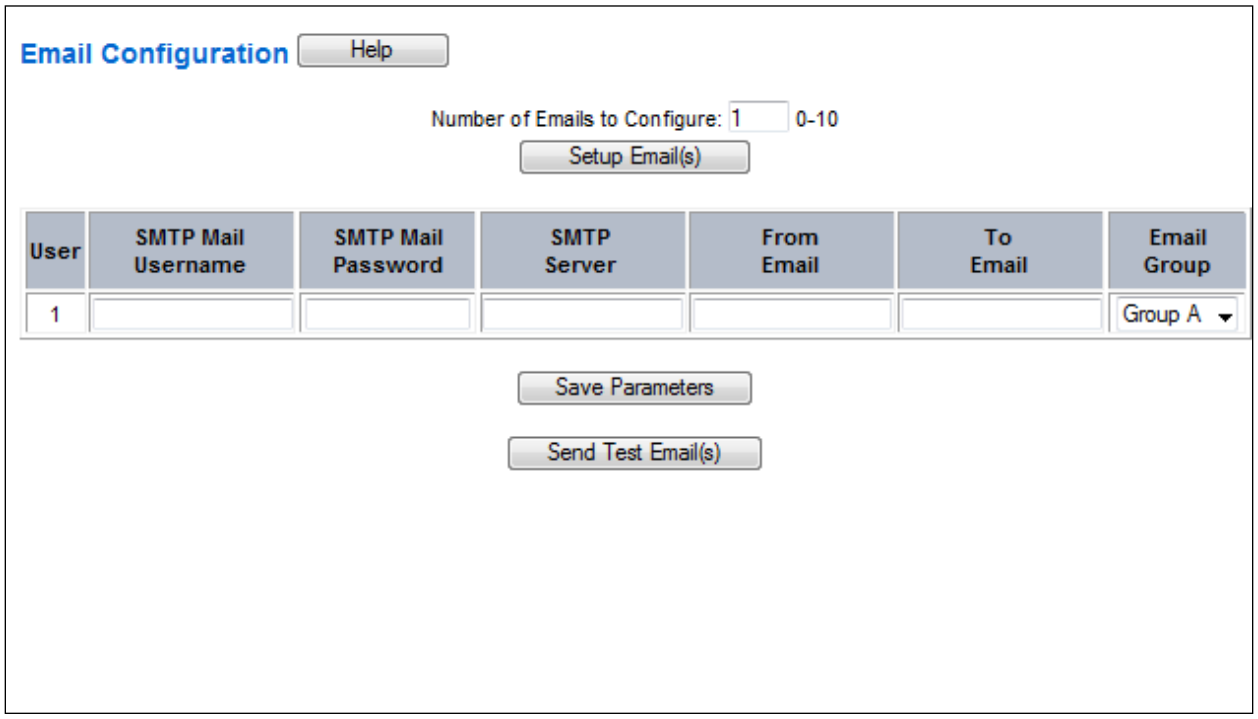
## Email Configuration

To setup e-mails on the 460 gateway, navigate to **Other->Email Configuration**.

You can configure up to 10 email addresses.

1) SMTP Mail Username: The email address that the SMTP server has set up to use.
2) SMTP Mail Password: If authentication is required, enter the SMTP Server's password (Optional).
3) SMTP Server: Enter the Name of the SMTP Server or the IP Address of the Server.
4) From E-mail: Enter the e-mail that will show up as the sender.
5) To E-mail: Enter the e-mail that is to receive the e-mail.
6) E-mail Group: Choose a group for the user. This is used in other web pages.

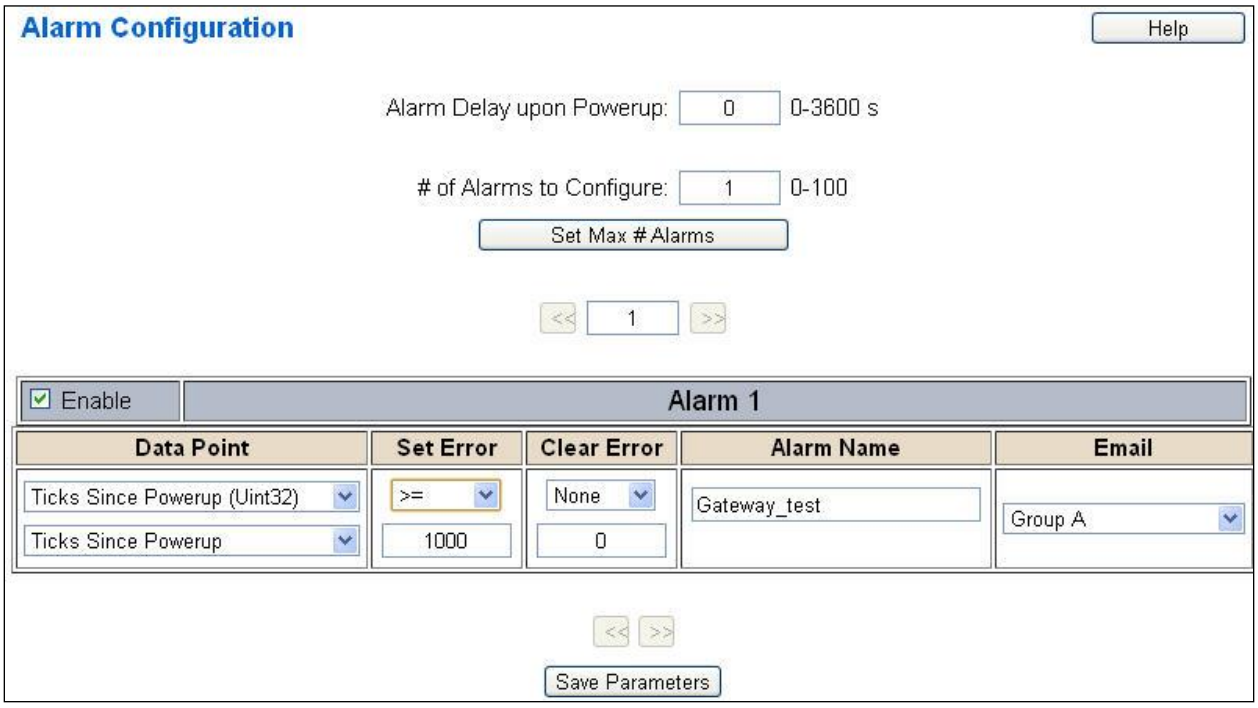
Click the **Save Parameters** button to commit the changes and reboot the gateway.



## Alarm Configuration

To setup alarms on the 460 gateway, navigate to **Other->Alarm Configuration**.

1) Alarm Delay upon Powerup: At Powerup, the gateway will have values of '0' stored for all data. This may cause alarms to trigger before these values are updated by the mating protocols. Set this field to provide needed time to update fields before considering values for alarms.




2) Enter the number of alarms to configure and click **Set Max # Alarms** to generate those lines.
3) In the Data Point Section:
    a. Top dropdown: select the Data Group. This dropdown menu will contain all groups that go from the gateway to the network.
    b. Lower dropdown: select the Data Point's Specific Point. This is used to select which point in the group will be monitored for alarms.
4) In the Set Error Section:
    a. Select the Set Error Operation in the top dropdown menu. Available options are <, >, <=, >=, !=, ==, and Change of State (COS). This is the operation that will be used to compare the Data Point value against the Error Value to determine if the alarm needs to be set.
    b. Select the Set Error Value. This value is used as: 'Data Point's Value' 'Operation' 'Value.' Ex: Ticks Since Powerup >= 1000. This will set the alarm after 1000 ticks have elapsed since the unit powered up.

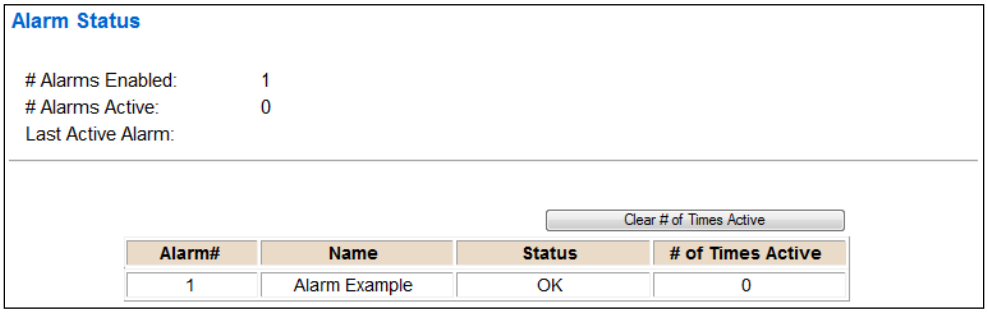
5) In the Clear Error Section:
    a. Select the Clear Error Operation. Available options are <, >, <=, >=, !=, ==, and Change of State (COS). This is the operation that will be used to compare the Data Point value against the Error Value to determine if the alarm needs to be cleared.
    b. Select the Clear Error Value.
    -Ex: Ticks Since Powerup >= 5000. This will clear the alarm after 5000 ticks have elapsed since the unit powered up.
6) Enter an Alarm Name. This will make the alarm unique and will be available in the Alarm Status page as well as in the email generated by the alarm.
7) Select an email to associate this alarm with. When an alarm is set, it sends an email. When an alarm is cleared, it will also send an email.

Click the **Save Parameters** button to commit the changes to memory and reboot the gateway.

## Diagnostics – Alarm Status

Alarm Status will only display under the Diagnostic menu tab if at least 1 Alarm is enabled.

1) # Alarms Enabled: This is a count of enabled alarms.
2) # Alarms Active: This is how many alarms are presently active (set).
3) Last Active Alarm: This is the last alarm that the gateway detected.
4) **Clear # of Times Active**: This will reset all alarms '# of Times Active' to 0.
5) Alarm #: The reference number to the given alarm on the alarm setup page.
6) Name: The name of the alarm.
7) Status: The current status of the alarm, either OK or ALARM.
8) # of Times Active: This count represents the number of times this alarm has become active. If an alarm is triggered, this count will increment.



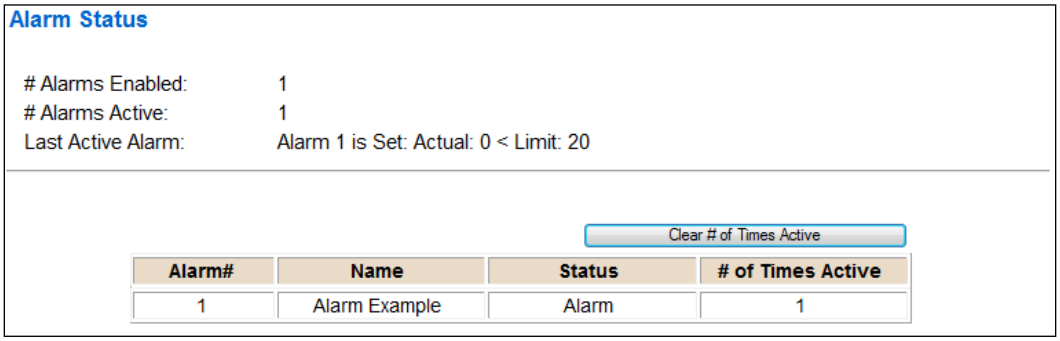

## Alarms – Active

While one or more alarms are active, every page will display 'Alarms Active' at the top of the page. This will no longer be displayed if all active alarms have been cleared.

When an alarm is activated, the following will occur:

1) A one-time notification will be sent out to the email associated with the alarm.
2) For duplicate emails to occur, the alarm must be cleared and then become active again.
3) *# Alarms Active* and *# of Times Active* will be incremented.
4) Status of the Individual Alarm will be set to *Alarm*.

5) *Last Active Alarm* field will be populated with details on what triggered the alarm.




## Alarms – Clear

When an alarm is cleared, the following will occur:

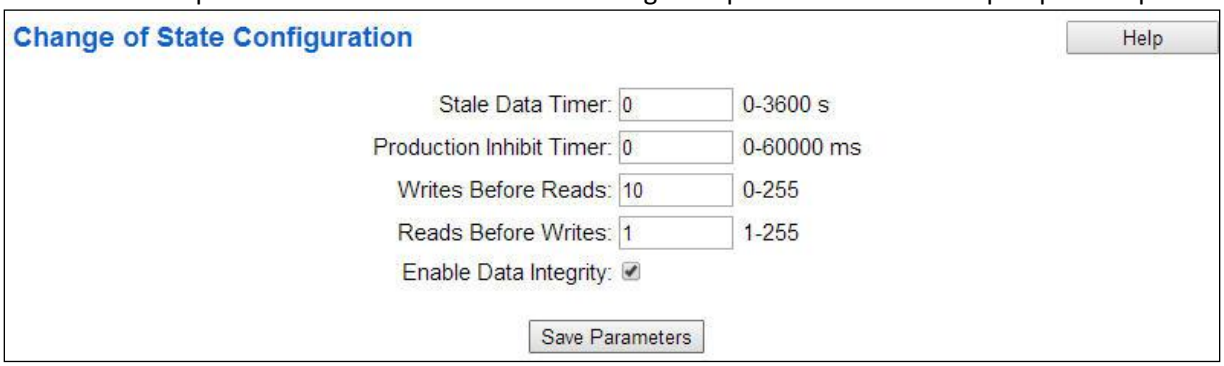
1) A one-time notification will be sent to the email associated with the alarm.
    a. For duplicate emails to occur, the alarm must become active and then be cleared again.
2) Total *# Alarms Active* will decrement. *Last Active Alarm* will not be changed.
3) Status of the Individual Alarm will be reset to *OK*.

## Change of State (COS) Configuration

To access the configuration files in the 460 gateway, navigate to dropdown **Other->COS Configuration**. The gateway, by default only writes when data has changed. The gateway also waits to write any data to the destination until the source protocol is successfully connected.

**Default values should fit most applications. Change these values with caution as they affect performance.**

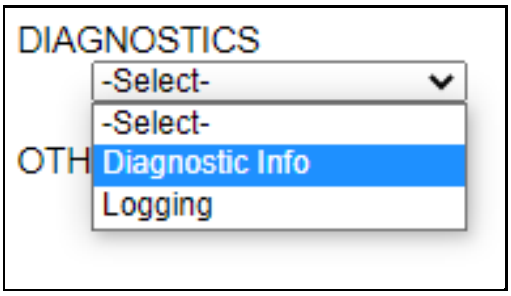
1) **Stale Data Timer**: If the data has not changed within the time allocated in this Stale Data Timer, the data will be marked as stale within the gateway and will force a write request to occur. This timer is to be used to force cyclic updates in the gateway, since data will only be written if it has changed by default. There is a separate timer per data mapping.
   **Gateway behavior**:
   - If time = 0s => (DEFAULT) The gateway will write out new values on a Change of State basis.
   - If time > 0s => The gateway will write out new values whenever the timer expires to force cyclic updates (write every *x* seconds).
2) **Production Inhibit Timer:** Amount of time after a Change of State write request has occurred before allowing a new Change of State to be written. This is to be used to prevent jitter. Default value is 0ms. This timer takes priority over the Stale Data Timer. There is a separate timer per data mapping. This timer is active only after the first write goes out and the first COS event occurs.
3) **Writes Before Reads**: If multiple writes are queued, execute # of Writes Before Reads before the next read occurs. Default is 10 and should fit most applications.
   **Warning:** A value of 0 here may starve reads if a lot of writes are queued. This may be useful in applications where a burst of writes may occur and you want to guarantee they all go out before the next set of reads begin.
4) **Reads Before Writes:** If multiple writes are queued, the # of Writes Before Reads will occur before starting the # of Reads Before Writes. Once the # of Reads Before Writes has occurred, the counter for both reads and write will be reset. Default is 1 and should fit most applications.
5) **Enable Data Integrity**: If enabled, do not execute any write requests to the destination until the source data point is connected and communicating. This prevents writes of 0 upon power up.




Click the **Save Parameters** button to commit the changes to memory and reboot the gateway.

## Diagnostics Info

The Diagnostics page is where you can view both protocols' diagnostics information, # of Data Mappings, # of String Mapping and # Alarm Mappings.




For protocol specific diagnostic information, refer to the next few pages.

## Diagnostics Mapping

This section displays the number of mappings that are enabled, Data Mapping and String Mapping will show the # of Errors and First Errors. Alarms will show # active and Last Alarm that was active.

**Common Errors:**

1) Destination or Source Point does not exist
   a) Solution: Re-map the mapping
2) Source or Destination Pointer too small
   a) There is not enough space on either the Source, or the Destination for the data you want to copy. This is typically seen when the Destination is smaller than the amount of data being transferred to it.
3) Range Discard, Min or Max Value
   a) The actual data value is outside of the defined range
4) Math Error
   a) Operation value cannot be 0
5) Scaling Error
   a) Source Min must be smaller than Source Max
   b) Destination Min must be smaller than Destination Max

| Data Mapping | |
|---|---|
| # Enabled: | 5 of 5 |
| # of Errors: | 0 |
| First Error: | |

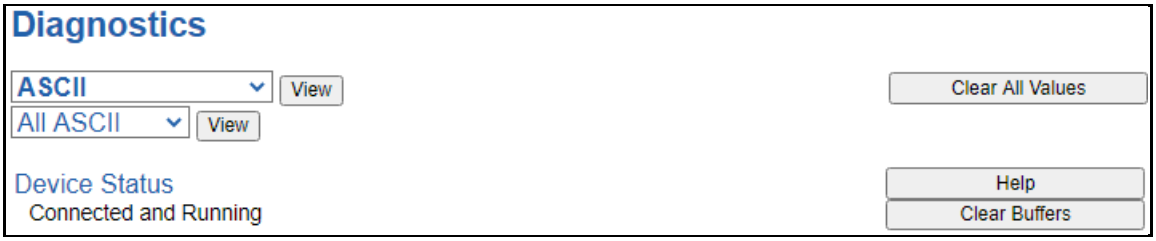
| String Mapping | |
|---|---|
| # Enabled: | 2 of 2 |
| # of Errors: | 0 |
| First Error: | |

| Alarms | |
|---|---|
| # Enabled: | 3 |
| # Active: | 0 |
| Last Active: | |

**Note:** you can also view this information on the Main Page.

## Diagnostics – ASCII

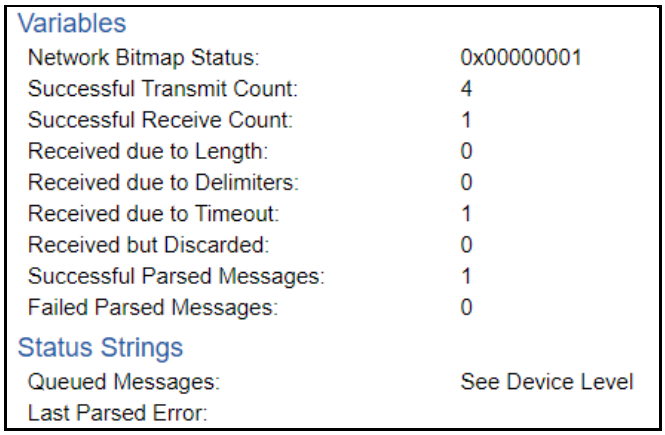
Select ASCII in the top dropdown menu on the Diagnostics Page to view a breakdown of the diagnostics that are displayed on the page. You may also view individual ASCII device counters and messages by selecting the device in the *All ASCII* dropdown and clicking **View**. Additional diagnostic information can be found by clicking the **Help** button.



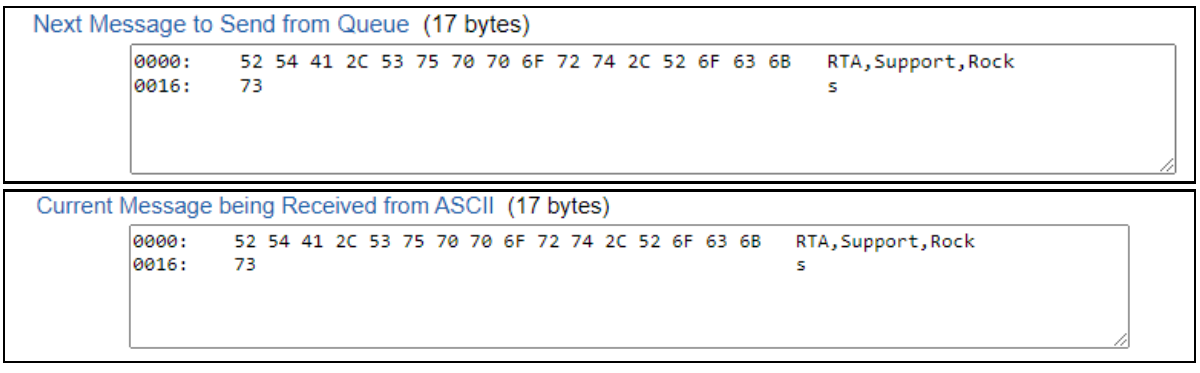

**NOTE**: This page will auto-refresh every five seconds with the latest data.

**Clear All Values** - This will only affect displayed values.

1) This will reset all displayed values back to zero and clear the Status Strings.
   Example: If viewing ASCII – Port #, this will only clear the values for Port #. This will reduce the *All ASCII* values indirectly.

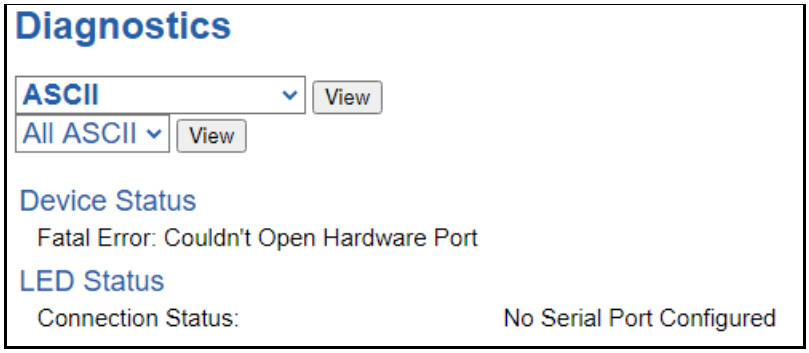



**Clear Buffers** - This will clear the Next Message to Send from Queue buffer and Current Message being received from ASCII buffer and any message stored in the Queue.

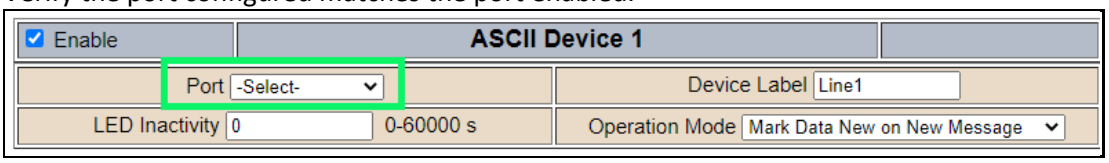


**Device Status** - This will only display when viewing *All ASCII*.

1) Connected and Running– The gateway is connected to all the ASCII devices and data is being received/transmitted.
2) Not Connected – There have been no messages received or transmitted.
   a. Verify that the serial /TCP/IP/USB settings match your device.
3) Fatal Error: Hardware Port Not Configured – The port selected on the ASCII Configuration page is not configured.

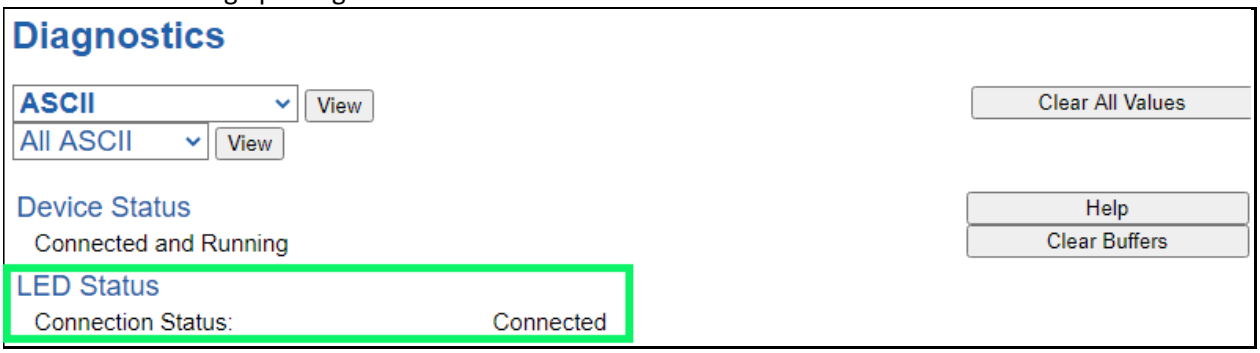



   a. Verify the ASCII device is enabled and configured.
   b. Verify the port configured matches the port enabled.



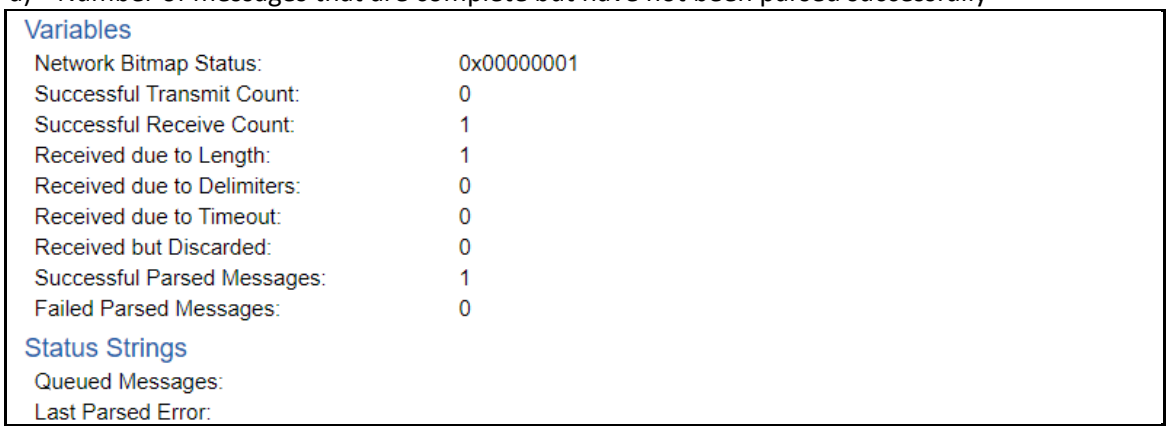

**LED Status** - This is the Status for *All ASCII* or the specific ASCII device selected.

1) Solid Green (Connected) – The gateway is receiving/transmitting data within the inactivity period for all the ASCII devices that are configured and enabled.
2) Flashing Green (Not Connected/First Time Scan) – Start up state. No messages have been received or transmitted, but port is connected.
3) Flashing Red (Connection Timeout) – The only way to exit this state is with a valid received message.
   a. Data has been discarded due to the queue being full.
   b. Data has not been received/transmitted within the inactivity period.
   c. Port not opened.
   d. Message parsing has failed.



**Variables** - These are the values for *All ASCII,* or the ASCII device selected.

1) Successful Transmit Count:
   a) Number of messages that the gateway has transmitted to the ASCII device
2) Successful Receive Count:
   a) Number of complete messages that the gateway has received from the ASCII device
3) Received due to Length:
   a) Number of messages completed due to the Max Message Length being reached
4) Received due to Delimiters:
   a) Number of messages completed due to the Start or End Delimiters being seen
5) Received due to Timeout:
   a) Number of messages completed due to the Receive Character Timeout being reached
6) Received but Discarded:
   a) Number of messages that are complete but discarded due to the queue being full
   b) Change the Gateway Hold Msg Timeout to be less than what you currently have set
7) Successful Parsed Messages:
   a) Number of messages that are complete and have been successfully parsed
8) Failed Parsed Messages:
   a) Number of messages that are complete but have not been parsed successfully




**Status Strings** - These are the values for *All ASCII,* or the ASCII device selected.

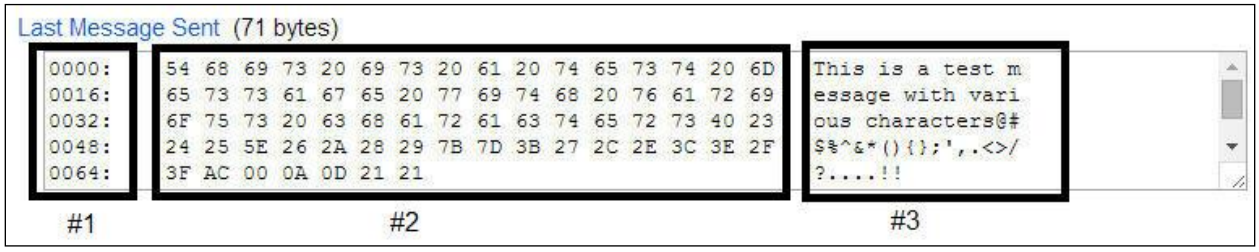
1) Queued Messages:
   a) The gateway will hold up to 20 (configurable) complete messages to send to the other protocol
   b) This will only increment if the Gateway Hold Msg Timeout is non-zero and messages are being received faster than we can send to the other protocol
2) Last Parsed Error:
   a) Last parsed error the gateway encountered

**Common Error Messages:**

1) **Number of Fields Invalid:** The total number of parsed fields is greater than the number of fields the gateway was expecting
2) **Discard:** The Field has been discarded
3) **Invalid Length for Field:** Number of characters parsed is greater than the number of characters that the gateway is expecting

4) **Calculated Length of Data exceeds 255 Characters:** Number of characters parsed within a field exceeds 255 characters

**Buffers**

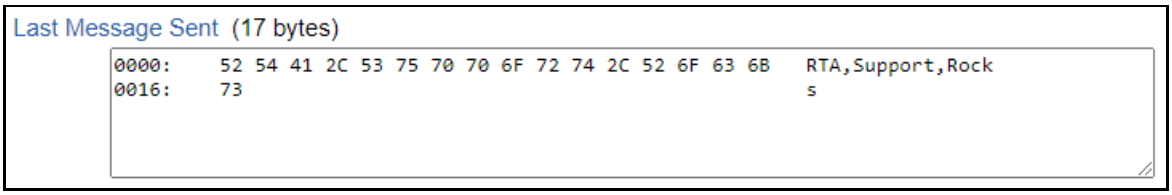



Each buffer text area is divided into three separate parts. Refer to screenshot above for labels.

1) Starting byte for that line
2) HEX character representation
3) ASCII character representation (Unprintable ASCII characters (like <CR>) will be displayed as '.' on the right-hand side of the buffer area.)

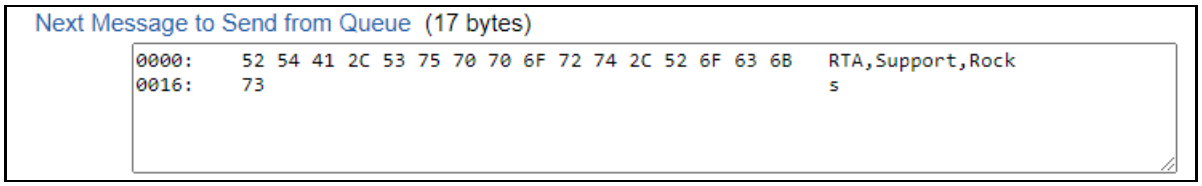
**Last Message Sent:**

1) Last complete message the gateway received from the ASCII device, processed, and sent to the other protocol



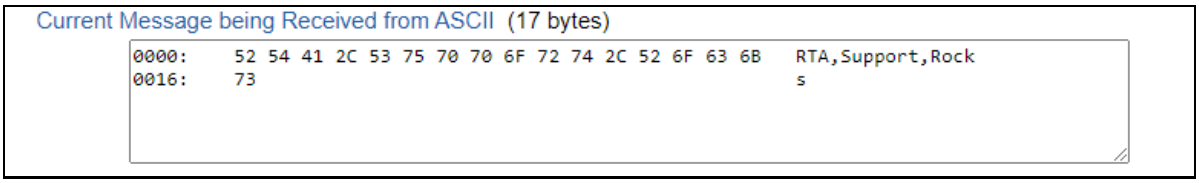

**Next Message to Send from Queue:**

2) This is the next complete message that the gateway has already received and processed from the ASCII device, and will be sent next to the other protocol




**Current Message being Received from ASCII:**

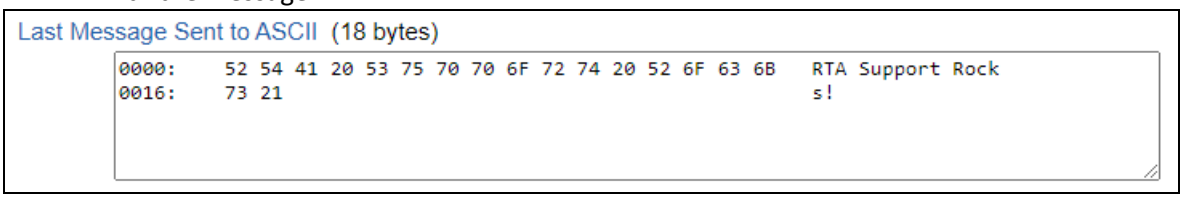
1) The current data that the gateway is receiving. The data needs to hit one of the three end cases to call a message complete:
   a) Max Message Length (max number of characters to receive)
   b) Receive Character Timeout (max time to wait after a character to call the current message complete)
   c) Delimiters (gateway has not received the delimiters that are specified)




**Last Message Sent to ASCII:**

1) Last message that the gateway sent to the ASCII device

   **Note**: The concatenated delimiters are not displayed in this message but will be transmitted with the message

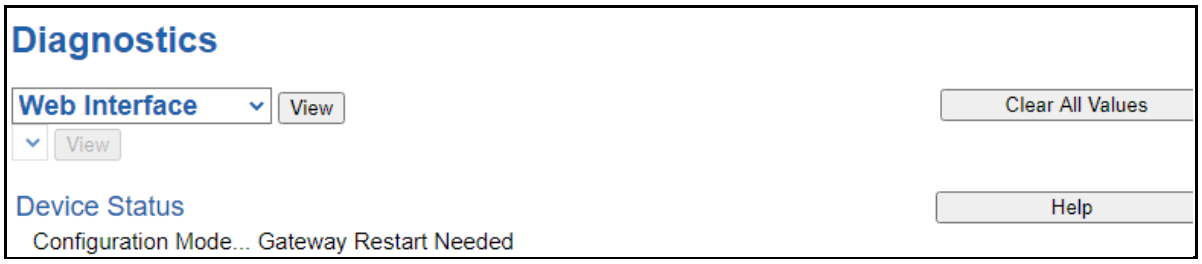



**Send Data from Gateway to ASCII:** (Used for testing only, Character limit of 1024):

1) Enter a message to send to your ASCII device
2) Can be used to test communication and test formatting of messages

## Diagnostics – Web Interface

Select **Web Interface** in the top dropdown menu on the Diagnostics Page to view a breakdown of the diagnostics that are displayed on the page.

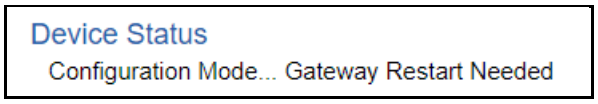



**NOTE**: This page will automatically refresh every five seconds with the latest data.

**Clear All Values** - This will only affect displayed values.

1) This will reset all displayed values back to zero and clear the Status Strings.
2) If the view is set to *Web Interface*, this will only clear the values for the Web Interface section of the gateway.

**Device Status:**

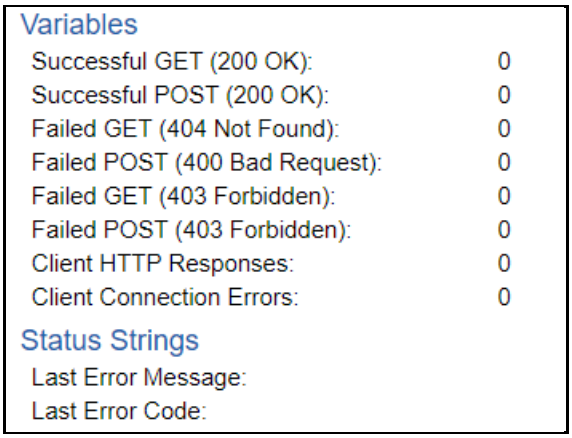



1) Connected and Running - The gateway is servicing HTTP GET or HTTP POST operations and the inactivity timeout (if configured) has not expired.
2) Connected (Idle) – The gateway is servicing HTTP GET or HTTP POST operations and the inactivity timeout has expired.
3) Not Connected – The gateway has never serviced any HTTP GET or HTTP POST operations.
4) Fatal Error: No Configuration – No data points have been configured for the Web Interface.

**LED Status**

1) Solid Green (Connected) – The gateway is servicing HTTP requests.
2) Flashing Green (First Time Scan) – Start up state. No HTTP requests have been processed, but data points are configured.
3) Flashing Red (Connected: Idle) – The gateway has not serviced a HTTP GET/POST operation within the Inactivity Period configured.
4) Solid Red (No Devices Configured/Enabled) – No data points are configured in the Web Interface.
5) Off (No Ethernet Cable) – The ethernet cable has been unplugged.

**Variables**




1) Successful GET (200 OK):
   a) Number of successfully serviced HTTP GET requests. In the case of a GET using a point name filter, at least one of the points in the list was found.
2) Successful POST (200 OK):
   a) Number of successfully serviced HTTP POST requests.
3) Failed GET (404 Not found):
   a) Number of messages where either the URL request was malformed or the requested group, device, or data point was not found in the gateway.
4) Failed POST (400 Bad Request):
   a) Number of failed HTTP POST operations due to:
5) A malformed x-www-form-urlencoded POST
6) Device or data point referenced does not exist in the gateway
7) Invalid data type for data point referenced
8) Failed GET (403 Forbidden):
   a) Number of times a GET request came from an unauthorized IP address or an IP address without enough READ privileges.
9) Failed POST (403 Forbidden):
   a) Number of times a POST request came from an unauthorized IP address or an IP address without enough WRITE privileges.
10) Client HTTP Responses:
   a) Number of times the *Automatic Data Transfer to User Host* gateway function successfully connected to the configured remote server and received an HTTP status code.
11) Client Connection Errors:
   a) Number of times the *Automatic Data Transfer to User Host* gateway function failed to connect to the configured remote server.

**Status Strings**

1) Last Error Message:
   a) Message details about the last error. See Common Error Messages below for more information.
2) Last Error Code:
   a) Last HTTP Error Code resulting from a GET or POST request

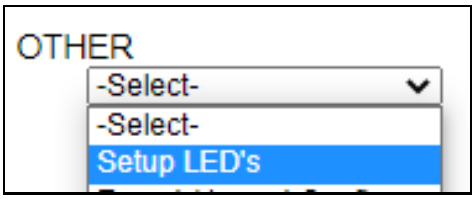
**Common Error Messages:**

The error message will alert if the message is for an XML or a JSON message. It will also indicate which line (x) and column (y) of the encoded data message contained the error and the reason why it triggered an error. Typically, the error message will also be followed by the client's IP address.

1) Server Error Strings
   a) Parsing Errors
      i) [XML/JSON] Parser (line: X col: Y): Invalid DEVICE Name
      ii) [XML/JSON] Parser (line: X col: Y): Invalid POINT Name
      iii) [XML/JSON] Parser (line: X col: Y): Over/Underflow for POINT Data
      iv) [XML/JSON] Parser (line: X col: Y): Invalid Data type for POINT
      v) [XML/JSON] Parser (line: X col: Y): [XML/JSON] Formatting Error
      vi) Invalid [XML/JSON] File Post Procedure
   b) Internal Errors – Should rarely see and indicated a major problem in the gateway
      i) [XML/JSON] Output I/O Failure
      ii) [XML/JSON] Output Buffer Overflow
      iii) Internal POST Error (500)
   c) HTTP Errors
      i) Error 404: URL Invalid.
   d) URL-Encoded POSTs – Will show "Invalid Post Attempt" followed by one of the following:
      i) POST to invalid Device: [Device Name]
      ii) Failed Write (Invalid Point Name) to [Device Name]
      iii) Failed Write (Overflow) to [Device Name] : [Point Name]
      iv) Failed Write (Invalid Data Type) to [Device Name] : [Point Name]
2) Client Error Strings
   a) HTTP Client Post rejected by server. Responded: xxx (xxx is an HTTP status other than 200)
   b) HTTP Client: Invalid Socket
   c) HTTP Client: Socket Closed
   d) HTTP Client: Socket Read Failed
   e) HTTP Client: Socket Read Timeout
   f) HTTP Client: Cannot Resolve Host
   g) HTTP Client: Cannot Connect
   h) HTTP Client: Invalid URL
   i) HTTP Client: Invalid Response
   j) HTTP Client: Proxy xxx

## LED Configuration

To modify the behavior of the LEDs on the 460 gateway, navigate to **Other->Setup LEDs**.




Each LED may be set to Disabled, Protocol 1, or Protocol 2. If either protocol is a master/client, you may set the LED to represent either all slaves/servers configured in the gateway or a slave/server device.

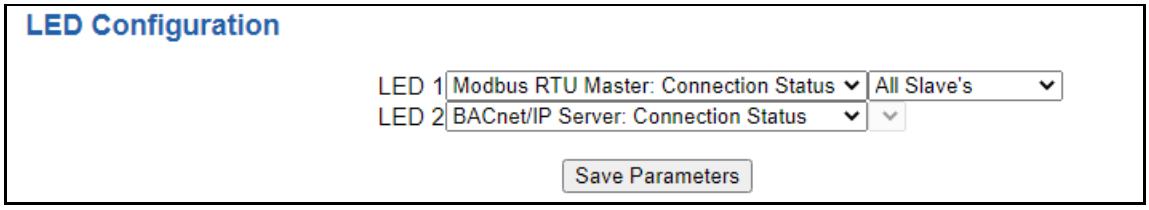
To select a slave/server device:

1) Select the protocol in the left dropdown menu.

2) Click **Save Parameters** to generate the second dropdown menu.

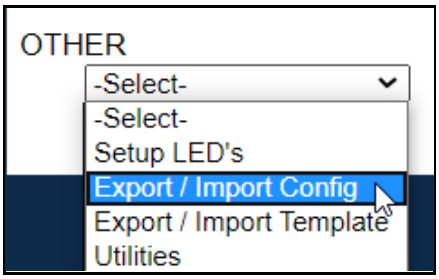
3) Select the individual slave/server in the right dropdown menu.

Click the **Save Parameters** button to commit the changes and reboot the gateway.

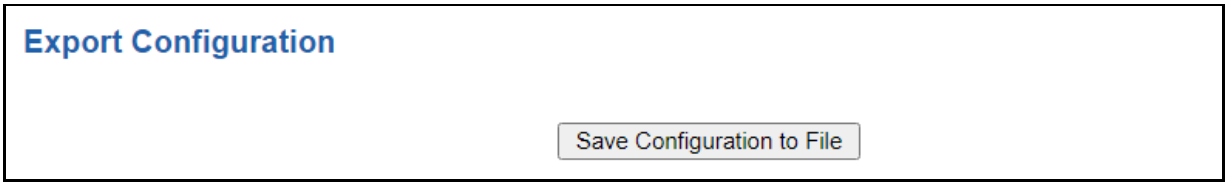


## Configuration Files

To access the configuration file in the 460 gateway, select the dropdown **Other->Export/Import Config**.




## Export Configuration




The Export Configuration allows you to save your configuration file for backup or to be imported into another gateway. This file is named *rta_cfg.rtax* by default.

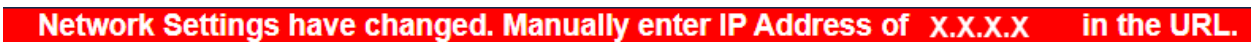
Upon clicking the **Save Configuration to File** button, you will be prompted to select a location to save the file. Different web browsers will yield different looks.

## Import Configuration

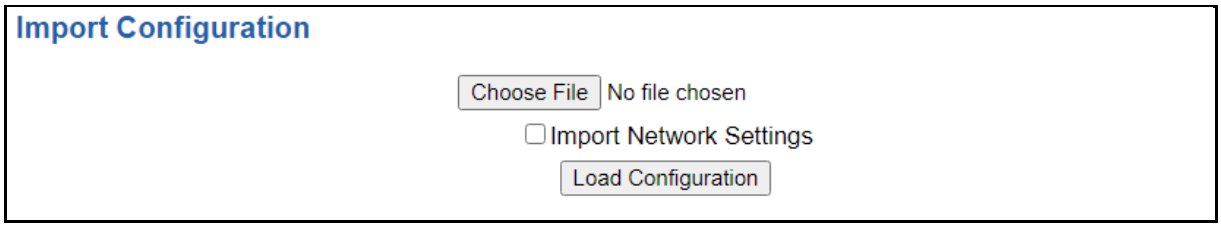
You can import a previously exported configuration file or a configuration file from another device into the 460 gateway, whenever it is in Configuration Mode.

Upon clicking the **Choose File** button, you will be prompted to select a location from which to load the saved file. Once the location is selected, you can choose the **Import Network Settings** checkbox if you want to load the network settings of the configuration file or just load the configuration without the network setting.

If you choose to Import Network Settings, this will override your current gateway's network setting with the settings in the configuration file. After you click on the Load Configuration button, a banner will display your gateway's new IP address.




If the configuration has successfully loaded, the gateway will indicate that it was successful, and a message will appear under the Load Configuration button indicating Restart Needed.




If it encountered an error while trying to load the saved configuration, the gateway will indicate the first error it found and a brief description about it under the Load Configuration button. Contact RTA Support with a screenshot of this error to further troubleshoot.

# Save and Replace Configuration Using SD Card

## Saving Configuration Using SD Card

This function saves the gateway's configuration automatically to an SD Card each time the gateway is rebooted via the **Restart Now** button on the web page. If this unit should fail in the future, the last configuration stored on the SD card and can be used for a new gateway to get the application back up and running quickly.

This SD Card replaces every configurable field in the gateway, **EXCEPT** for IP Address, Subnet Mask, and Default Gateway.

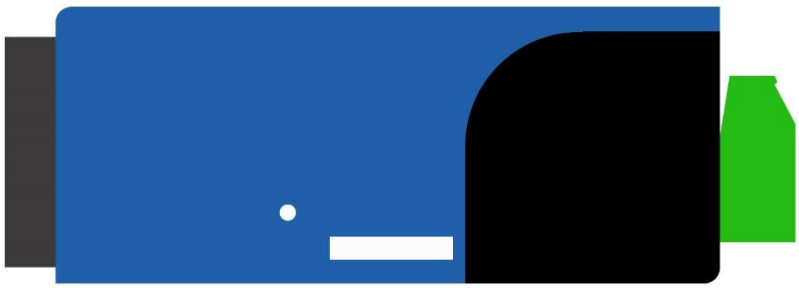
## Replacing Configuration Using SD Card

To replace a configuration in a gateway using the SD Card, a specific sequence of events must be followed for the replacement to happen correctly:

1) Extract SD Card from gateway you wish to copy the configuration from.
2) Power up the gateway you wish to copy the configuration to. DO NOT INSERT SD CARD YET.
3) Navigate to the webpage inside the unit.
4) Navigate to the dropdown **Other->Utilities**.
5) If you are not currently in *Mode: Configuration*, go into Configuration Mode by clicking the **Configuration Mode** button at the top left-hand side of the screen.
6) Press the **Revert to Manufacturing Defaults** button on the Utilities Page. The Configuration will ONLY be replaced by the SD Card if the gateway does not have a configuration already in it.
7) When the unit comes back in *Mode: Running*, insert the SD Card.
8) Do a hard power cycle to the unit by unplugging power. DO NOT RESET POWER VIA WEB PAGES.
    a. It will take an additional 30 seconds for the unit to power up while it is transferring the configuration. During this time, the gateway cannot be accessed via the web page.
9) When the unit comes back up, the configuration should be exactly what was on the SD Card.

## Intelligent Reset Button

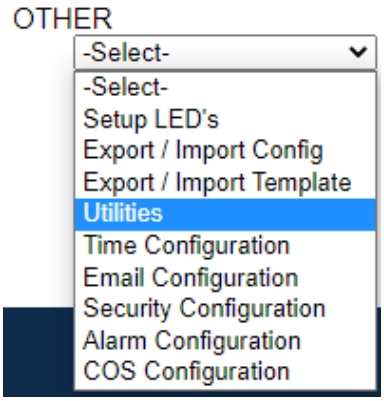
If the IP Address of the gateway is forgotten or is unknown, there is an easy way to recover the IP Address using a reset button on the hardware.

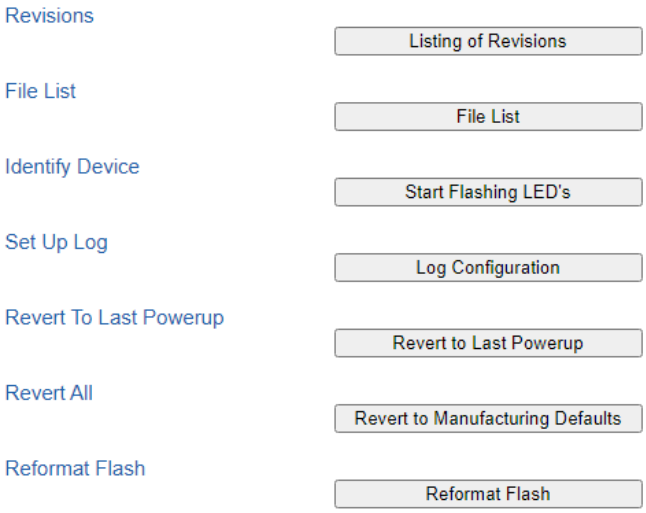


1) On the side of the gateway with the SD card slot, there is a small pinhole. Using a paperclip, press the button through this pinhole and hold the button for at least 5 seconds.
2) After 5 seconds, the unit will acknowledge the command and LED 1 and LED 2 will start an alternate Blink Green quickly pattern.
3) Release the button and the gateway will reset to default IP settings (DHCP).

## Utilities

To access the Utilities page in the 460 gateway, navigate to **Other->Utilities**. The Utilities screen displays information about the gateway including Operation Time, File System Usage, Memory Usage, and Memory Block Usage.




Here you can also:

- View the full revision of the software.
- View all the files stored in the Flash File System within the gateway.
- Identify your device by clicking the **Start Flashing LEDs** button. By clicking this button, the two diagnostic LEDs will flash red and green. Once you have identified which device you are working with, click the button again to put the LEDs back into running mode.
- Configure the size of the log through the Log Configuration.
- Bring the device back to its last power up settings.
- Bring the device back to its original manufacturing defaults.
- Remove the Configuration File and Flash Files within the gateway.