

User Guide

Model VP6-1460
ValuPoint IoT
Edge Server for Modbus

Rev. 1.0 – Feb. 2024

© 2024 Control Solutions, Inc.

VP6-1460 User Guide Contents

[1 Introduction](#)

- 1.1 How to Use This Guide
- 1.2 Important Safety Notice
- 1.3 Warranty

[2 Connecting the ValuPoint](#)

- 2.1 Overview of ValuPoint Operation
 - 2.1.1 Cloud Based Application
 - 2.1.2 Stand-Alone Application
- 2.2 Where to Start
- 2.3 Connectors and Indicators
- 2.4 Web User Interface
- 2.5 External Programming Tool

[3 System Configuration and Resources](#)

- 3.1 Using the File Manager
 - 3.1.1 Load, Save, Create XML Configuration File
 - 3.1.2 Select Startup Configuration
 - 3.1.3 Delete a File
 - 3.1.4 Import CSV File
 - 3.1.5 Clear Configuration
- 3.2 Configuration Files and Restoring Default Settings
- 3.3 Network Configuration
 - 3.3.1 IPv4, IPv6 Settings
 - 3.3.2 NTP Time Server Settings
 - 3.3.3 Port Settings
- 3.4 Resource Allocation
- 3.5 User Login Passwords

[4 Configuring Local Registers](#)

- 4.1 Creating Local Registers
- 4.2 Configuring Physical I/O
- 4.3 Special Features of Local Registers
- 4.4 Local Register Calculate Rules
- 4.5 Local Register Copy Rules
- 4.6 Device Status Reporting

[5 Configuring ValuPoint as a Modbus RTU Master](#)

- 5.1 Modbus RTU Device Configuration
- 5.2 Modbus RTU Master Read Maps
- 5.3 Modbus RTU Master Write Maps
- 5.4 Modbus RTU Master Data Displayed by Slave
- 5.5 Modbus RTU Errors

[6 Configuring ValuPoint as a Modbus TCP Client](#)

- 6.1 Modbus TCP Device Configuration
- 6.2 Modbus TCP Client Read Maps
- 6.3 Modbus TCP Client Write Maps
- 6.4 Modbus TCP Client Data Displayed by Server
- 6.5 Modbus TCP Errors

[7 Configuring ValuPoint as a Modbus RTU Slave](#)

- 7.1 Modbus RTU Device Configuration
- 7.2 Modbus RTU Slave Register Map
- 7.3 Modbus RTU Slave Diagnostic

[8 Configuring ValuPoint as a Modbus TCP Server](#)

- 8.1 Modbus TCP Device Configuration
- 8.2 Modbus TCP Server Register Map

8.3 Modbus TCP Server Diagnostic

[9 Configuring Event Rules](#)

- 9.1 Event Rule List
- 9.2 Event Rule Details

[10 Configuring Email Client & Notifications](#)

- 10.1 Assigning Email Templates
- 10.2 Editing Email Templates
- 10.3 Email Recipients
- 10.4 Email Server
- 10.5 Using Gmail

[11 Configuring Local Data Logging](#)

- 11.1 Selection of Data Points
- 11.2 Log Rate and File Send
- 11.3 CSV File Format
- 11.4 Anticipated File Size

[12 Configuring the Scheduler](#)

- 12.1 Weekly Schedule
- 12.2 On Demand Scheduled Events
- 12.3 Holidays
- 12.4 Astronomical Clock

[13 Configuring the IoT Client](#)

- 13.1 Thing Points or Attributes
- 13.2 Thing ID and Subscribe Topics
- 13.3 Thing Files
- 13.4 Thing Status
- 13.5 Testing Thing's Connection

[14 Configuring IoT Client to Publish to AWS](#)

- 14.1 Create and Register a "Thing"
- 14.2 Update Policy
- 14.3 Configure IoT Client
- 14.4 Test Publish to Device Shadow

[15 Configuring IoT Client to Subscribe to AWS](#)

- 15.1 Configure IoT Client
- 15.2 Use MQTT Test Client to Test Subscription

[16 Using Mosquitto MQTT](#)

- 16.1 Configuring the Mosquitto Linux Server
- 16.2 Publishing Thing Points to Mosquitto

[17 Using Thingsboard.io MQTT](#)

- 17.1 Introduction to Thingsboard
- 17.2 Adding a Device
- 17.3 Creating a Dashboard
- 17.4 Adding a Sensor Widget
- 17.5 Adding an Actuator Widget
- 17.6 Diagnostics

[18 Programming with Script Basic](#)

- 18.1 Creating a Program
- 18.2 Testing the Program
- 18.3 Setting the Program to Auto-Run on Startup
- 18.4 Special Functions
 - 18.4.1 Serial Port Functions
 - 18.4.2 Register Access
 - 18.4.3 Register New-Data Status
 - 18.4.4 Register Access Status
 - 18.4.5 LED Control via Phantom Register Access
 - 18.4.6 Program Watchdog Timer
 - 18.4.7 Error Information Retrieval
 - 18.4.8 IP Address Retrieval
 - 18.4.9 Free Memory
- 18.5 Example: Program Triggering of Publish to AWS

- 18.6 Example: Writing to an Alphanumeric Display
- 18.7 Example: Device and Rule Diagnostics
- 18.8 Example: Connecting Custom Serial Device to AWS

19 User HTML

- 19.1 Static HTML
- 19.2 Dynamic Data Tags in User HTML
- 19.3 Live JavaScript Gauges

20 REST API

- 20.1 GET Data from Device
- 20.2 POST Data to Device
- 20.3 Raw Data Query

Appendix A Hardware Details

- A.1 Wiring, Physical I/O Connections
- A.2 Front Panel LED Indicators
- A.3 RS-485 Line Termination
- A.4 Soft Configuration Reset
- A.5 Discovering Lost IP Address
- A.6 Forced Hard Configuration Reset
- A.7 Firmware Update Notes
- A.8 Battery Replacement

Appendix B Modbus CSV Import Files

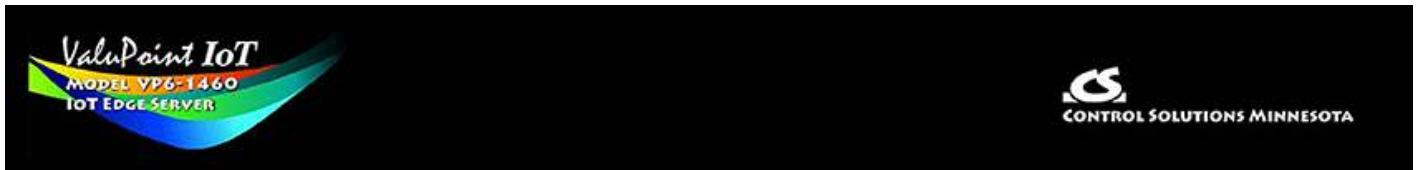
- B.1 Modbus RTU Master Read/Write Maps
- B.2 Modbus TCP Client Read/Write Maps
- B.3 Register Types
- B.4 Register Formats

Appendix C Trouble Shooting

- C.1 Modbus RTU Trouble Shooting
- C.2 Modbus TCP Trouble Shooting
- C.3 AWS Trouble Shooting
- C.4 Modbus Reference Information

Appendix D SSL Certificates for Secure Web (HTTPS)

- D.1 X.509 Auto-Certificate Generation
- D.2 External Certificates
- D.3 Certificate Generation Script (Linux)



1. Introduction

1.1 How to Use This Guide

This user guide provides background information on how the ValuPoint works, and an overview of the configuration process. There are several sections for groups of tabs found in the web interface in the ValuPoint which is accessed by opening a web browser and browsing to the IP address of the device.

You should at least read Sections 1 and 2 to gain an understanding of how the ValuPoint functions. You can use Sections 3 through 12 as reference material to look up as needed. You will need to read sections beginning at Section 13 to start to understand how to connect to the Amazon web servers. There is a "Quick Help" section at the bottom of each web page in the ValuPoint which is generally sufficient for quick reference in setting up the ValuPoint.

1.2 Important Safety Notice

Proper system design is required for reliable and safe operation of distributed control systems incorporating any Control Solutions product. It is extremely important for the user and system designer to consider the effects of loss of power, loss of communications, and failure of components in the design of any monitoring or control application. This is especially important where the potential for property damage, personal injury, or loss of life may exist. By using ANY Control Solutions, Inc., product, the user has agreed to assume all risk and responsibility for proper system design as well as any consequence for improper system design.

CAUTION: The lithium battery contained in this device may explode if mistreated. DO NOT recharge, disassemble, or dispose of in fire.

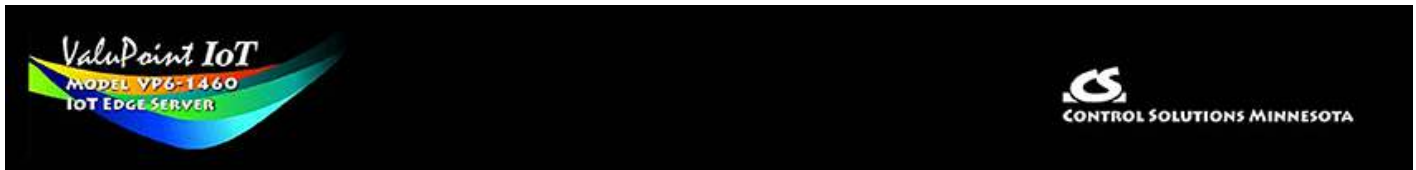
No action is required of the user to activate the battery that backs up the real time clock. Important: Replace battery with BR1225A only. Use of another battery may present a risk of fire or explosion.

1.3 Warranty

This documentation is provided "as is," without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of fitness or merchantability for a particular purpose. Control Solutions may make improvements and/or changes in this documentation at any time. This documentation could include technical inaccuracies, typographical errors, and the like. Changes are periodically made to the information herein; these changes may be made without notice.

Product Warranty: All Control Solutions products are warranted against defects in materials and workmanship for a period of time from date of shipment from factory as follows: Two years on non-mechanical parts, one year on mechanical parts (e.g. relays). Defective units will be repaired or replaced, at manufacturer's discretion, at no cost to user except when negligence or improper use has resulted in damage. The express warranty stated herein is in lieu of all other warranties, express or implied, including without limitation any warranties of merchantability or fitness for a particular purpose and all other warranties are hereby disclaimed and excluded by Control Solutions, Inc.

Configuration errors made by customer are not covered under warranty. Damage caused by incorrect electrical connection is not covered under warranty. Removing circuit boards from their enclosures will void the warranty - the complete product with all of its original circuit boards and components must be returned for warranty consideration.



2. Connecting the ValuPoint

2.1 Overview of ValuPoint Operation

The ValuPoint connects physical I/O (e.g. sensors) to the Internet. In addition, the VP6-1460 turns any Modbus device into a Thing on the Internet of Things. Gain instant access to a wide range of machine learning and AI capabilities, a wide range of cloud based data storage and analytics, and a variety of cloud driven event handling and notification capabilities. All of this is made possible by the ValuPoint IoT Edge Server and the many features of Thingsboard.io, Amazon Web Services, or generic Mosquitto MQTT.

Are you not a fan of the "cloud"? No problem. The ValuPoint IoT Edge Server can provide you with many of the same capabilities on its own, without any cloud. You can take advantage of local data logging, local email client, and event notifications generated within just the IoT Server itself.

An IoT Device typically has one or more of these functions:

- Monitoring something and collecting data for later analysis
- Controlling something according to some given algorithm or schedule
- Notifying somebody when something goes wrong

All of these functions are supported both for cloud based implementation and stand-alone implementation using the ValuPoint IoT Edge Server.

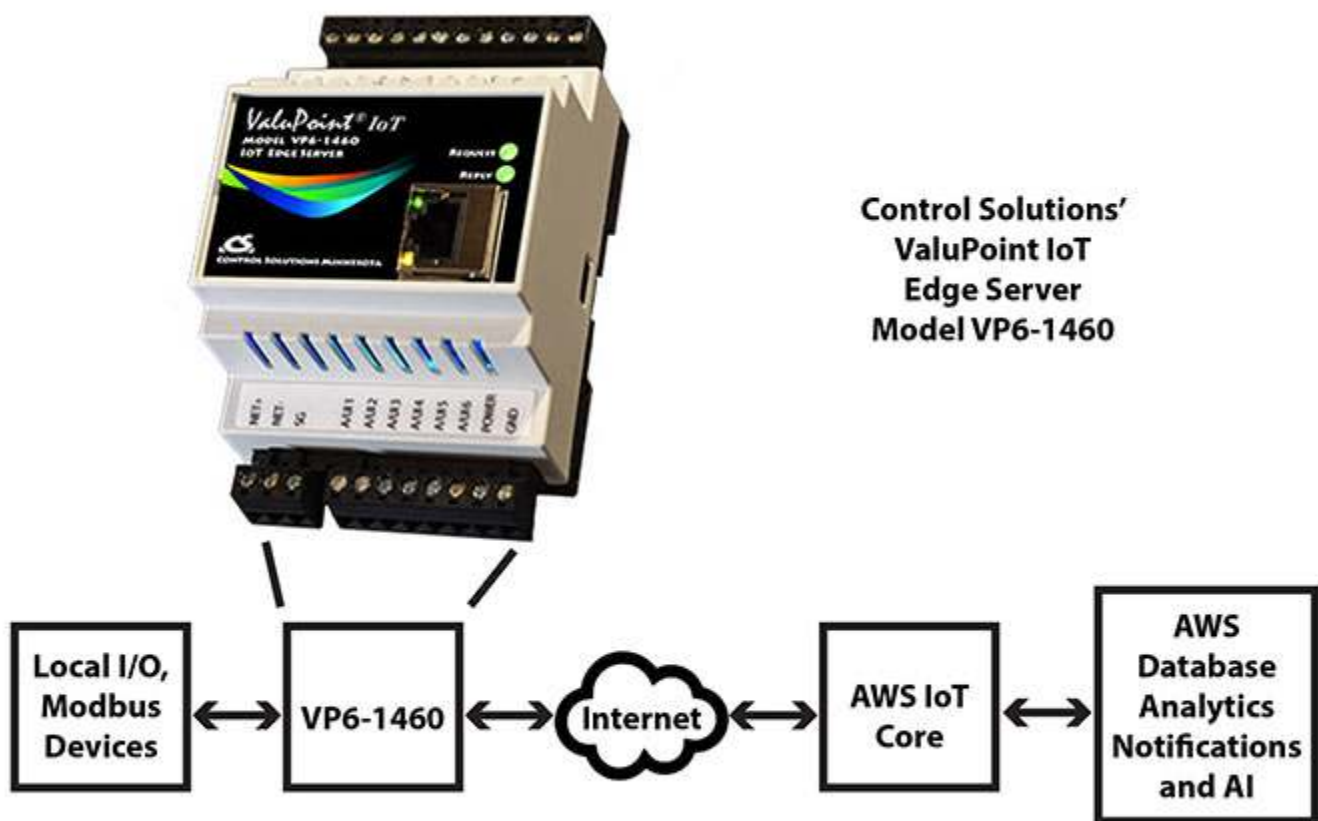
Hardware Features of Model VP6-1460

- 12 Analog/universal inputs, software selectable types
 - 0-10VDC, thermistor, discrete, dry contact, pulse
 - 0.1% reference, 12-bit resolution
 - Non-volatile totalizing count inputs (to 2Hz on all channels, to 1kHz on 4 channels)
- 2 Discrete outputs
 - Form A relay
 - 2A @ 120VAC
 - 2A @ 30VDC
- Battery backed real time clock/calendar
- Modbus TCP over Ethernet 10/100BaseT
- Isolated RS-485 port, Modbus RTU at 1200 to 115200 baud
- Powered by 18-30VDC or 24VAC 50/60 Hz Class 2, 0.3A max.
- DIN rail mounting, 100mm H x 70mm W x 60mm D
- Operating temperature -40°C to +80°C; Humidity 5% to 90%

- FCC Class A, CE Mark
- Listed to UL 916 and (Canadian) C22.2 No. 205-M1983

2.1.1 Cloud Based Application

Control Solutions chose to provide direct integration with Amazon Web Services simply because AWS offers the widest array of available capabilities at the best price. These capabilities have been scaled for large applications and are used by large corporations. At the same time, these capabilities are readily accessible for the much smaller enterprise with just a few devices to monitor. In fact, Amazon Web Services are affordable for using with just one device, unlike many of the IoT or MQTT enterprise solutions.



The ValuPoint will continuously poll its sensor inputs, and if configured to do so, poll one or more Modbus RTU and/or Modbus TCP devices, collecting data from the list of registers you provide. Based on rules you create, the ValuPoint will decide if and when to publish that data to the AWS server or other MQTT broker. You can also configure the ValuPoint to subscribe to data coming from the AWS server or other MQTT broker, which you can then write out to local outputs or to Modbus devices to manage setpoints and the like.

AWS IoT is based on the MQTT protocol. Sending data to the AWS server and receiving data from the AWS server is all done in MQTT protocol using JSON to represent the

data.

Examples of JSON formatted MQTT messages are as follows:

**MQTT message from device
to AWS server:**

```
{
  "state": {
    "reported": {
      "csiSensor1": 70,
      "csiSensor2": 68
    }
  }
}
```

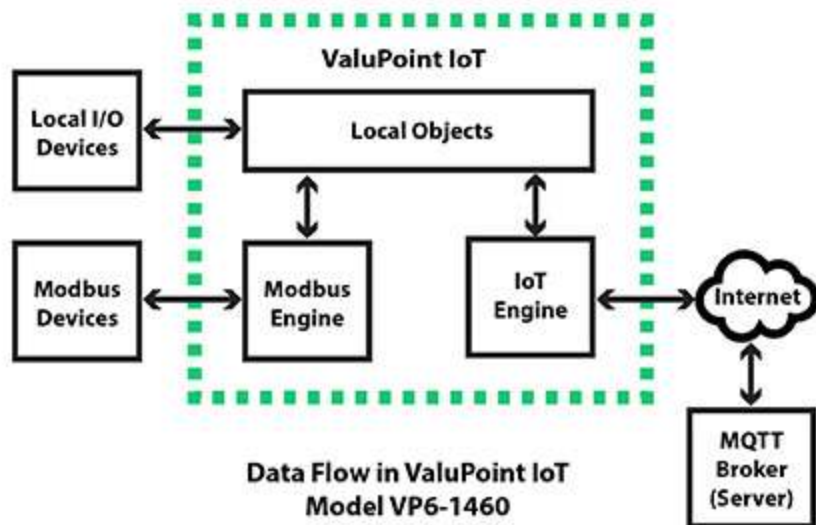
**MQTT message from AWS server
to device:**

```
{
  "state": {
    "desired": {
      "csiActuator1": 50
    }
  }
}
```

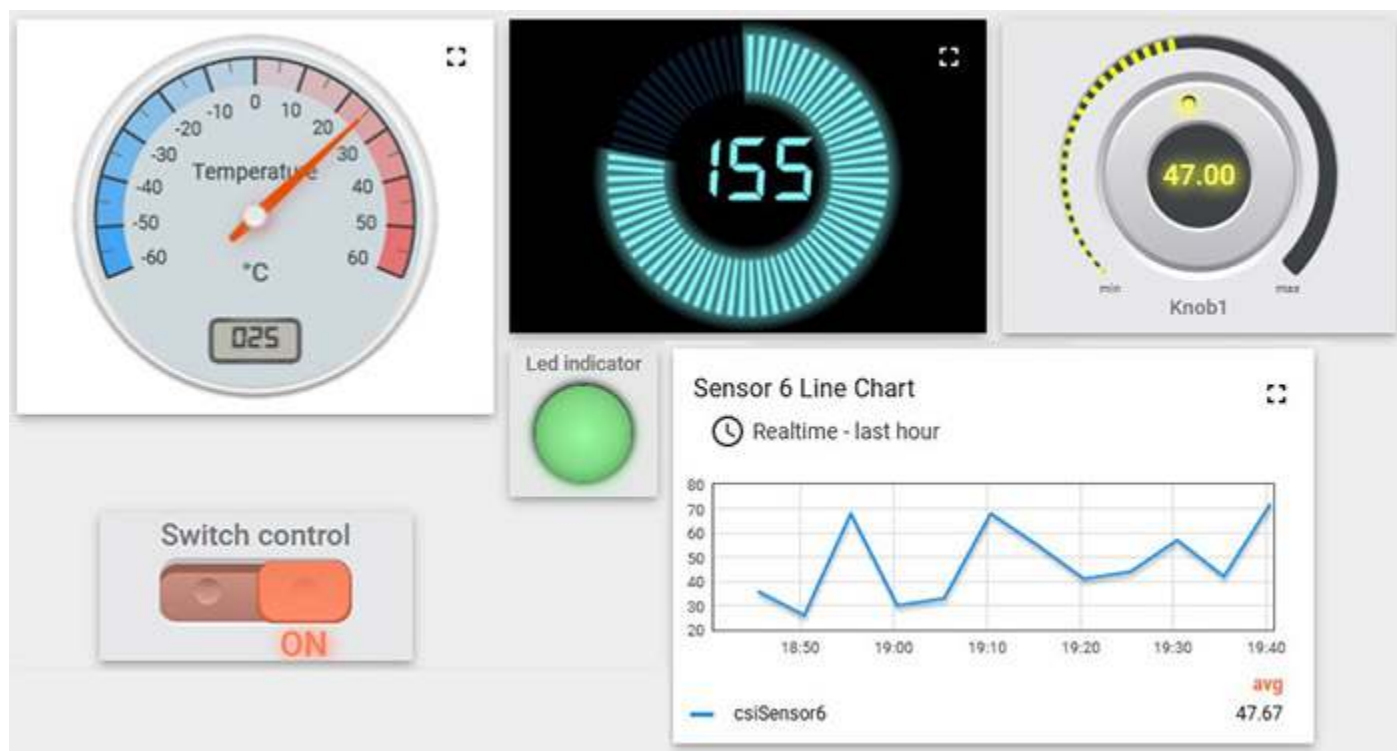
The MQTT "publish" action, in controls terms, is most closely associated with sensors. Your hardware has collected sensor data, and you want to send that sensor data to a server or to other control devices. To send that data, you "publish" it.

The MQTT "subscribe" action, in controls terms, is most closely associated with actuators. The "subscribe" action would also be associated with control setpoints. You can never force data into a device via MQTT. The device, in our case the ValuPoint, must subscribe to the source of data effectively asking to be informed of changes. Once you have subscribed to an MQTT source of data, then when received, you can use that data to control actuators or update setpoints.

The data flow in the ValuPoint is illustrated below. Data is collected from local I/O or collected from Modbus devices by the Modbus engine which stores that data in its local registers or data objects. Modbus is collected according to a set of rules or "maps" created by the user. The local data is automatically updated on a continual basis. Meanwhile, the IoT engine is looking at the data and its set of publish and subscribe rules to decide when to publish data from the local registers to the AWS server or other MQTT broker. These rules are created by the user and data will be published according to the criteria set up by the user (you).



The ValuPoint IoT Edge Server includes support for Thingsboard.io MQTT services. Thingsboard provides a number of capabilities including interactive real time dashboards with widgets such as gauges, knobs, buttons, and charts. Here is an example of a demo dashboard we built for test purposes.

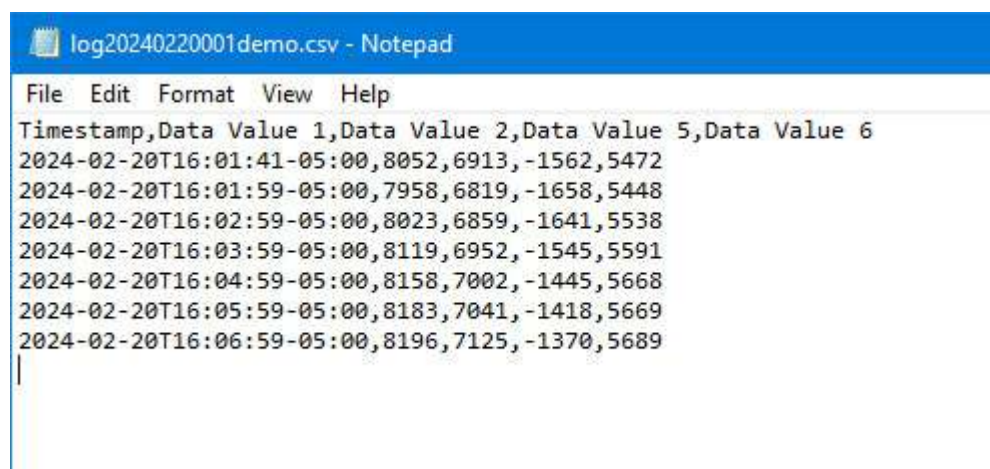


The ValuPoint is capable of more than just transferring data between local I/O or Modbus devices and the AWS servers or other MQTT brokers. It includes Script Basic built in to provide easy-to-use local programming for data analysis and local decision making. This capability is referred to as Edge Computing in IoT terminology.

2.1.2 Stand-Alone Application

The ValuPoint supports stand-alone data logging by logging selected data points to a

local file in CSV format, and then automatically emailing that file to you from time to time. Once received, you can do anything with that data that you can normally do with any standard spread sheet program.

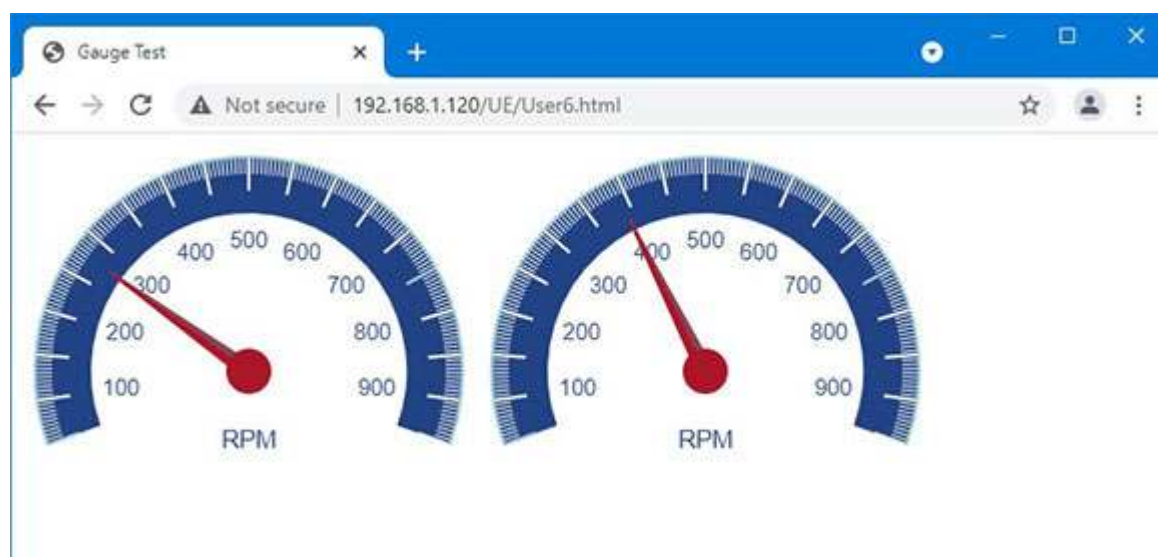


```
log20240220001demo.csv - Notepad
File Edit Format View Help
Timestamp,Data Value 1,Data Value 2,Data Value 5,Data Value 6
2024-02-20T16:01:41-05:00,8052,6913,-1562,5472
2024-02-20T16:01:59-05:00,7958,6819,-1658,5448
2024-02-20T16:02:59-05:00,8023,6859,-1641,5538
2024-02-20T16:03:59-05:00,8119,6952,-1545,5591
2024-02-20T16:04:59-05:00,8158,7002,-1445,5668
2024-02-20T16:05:59-05:00,8183,7041,-1418,5669
2024-02-20T16:06:59-05:00,8196,7125,-1370,5689
```

There are several features available for supporting control algorithms, and a real time scheduler is included in the ValuPoint. Direct response control can sometimes be handled with just an event rule. You can use a sequence of Calculate instructions for simple algorithms. For complex requirements, you can write a totally custom program using the Script Basic support included in the ValuPoint.

Notifications generated locally use event rules to detect when a condition exists. The local email client will email you or others to let you know what is going on.

The ValuPoint supports a customizable user interface so that you can have your own version of a "dashboard" for your device. With a little programming and a little help from JavaScript, you can display live gauges in your web browser.



2.2 Where to Start

- Start by connecting the ValuPoint as noted in the following section. Then set the IP address of your ValuPoint, and get familiar with the File Manager. You will find these covered in Section 3.

- Create some registers so you have a place to put data. Section 4 talks about this.
- Decide how you're going to talk to your Modbus device. Are you using Modbus TCP or RTU? Should the ValuPoint be master or slave? Based on how you answer these questions, you will choose from Sections 5 through 8.
- Are you interested in using Amazon Web Services? If so, skip to section 13.
- Are you interested in local alarm monitoring? Event rules are where you will tell the ValuPoint what you want to watch for. These are covered in Section 9, and setting up the local email client to send you a notification about these events is covered in Section 10.
- Are you interested in local data logging? This is covered in Section 11.
- Are you interested in scheduling things that you want to happen? Take a look at the scheduler in Section 12.
- To get rolling with Amazon Web Services, start at Section 13 where we talk about setting up the ValuPoint to talk to AWS. Then move on to Sections 14 and 15 to cover the MQTT subscribe and publish configuration.

For using AWS, you will need to create an account at <https://aws.amazon.com> if you haven't already. Once there, you will find a seemingly endless source of documentation on AWS IoT as well as the many other related services available to you via AWS.

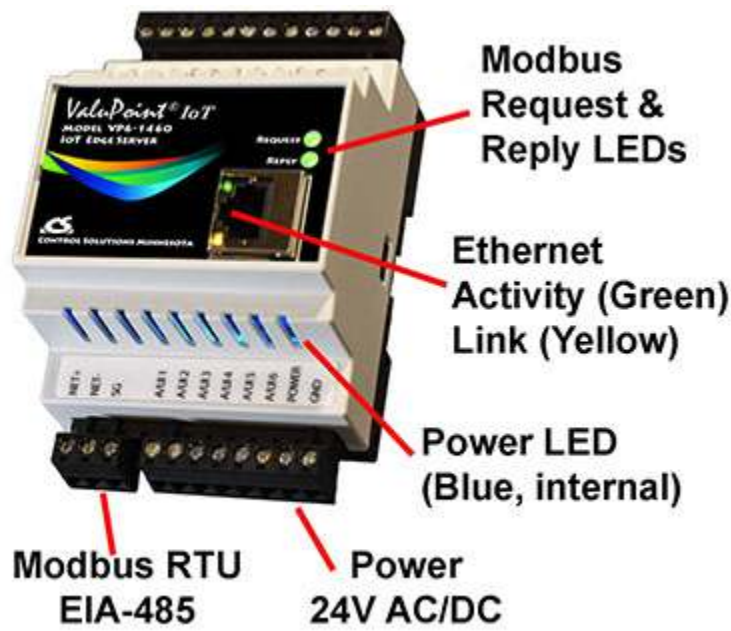
- You can install the open source Mosquitto MQTT broker on your own server. If you wish to use Mosquitto MQTT, refer to Section 16 to get started.
- Thingsboard.io offers a graphical dashboard for MQTT devices that is relatively straight forward to set up. Refer to Section 17 to get started with Thingsboard.
- Local programming with Script Basic can apply to either cloud or non-cloud based applications. If you can't find a template that does what you want, you can probably write a program to do it. The local programming support is covered in Section 18.
- Last but not least are a couple of advanced topics. If you want to create your own custom web pages to be served by the ValuPoint's internal server, that is covered in Section 19. If you have an external server with an application that can use a REST API to query devices, the REST API is covered in section 20.
- Various details are covered in Appendix A through D. Be sure to look at the first 3 sections of Appendix A which cover hardware details you will need to be aware of. You can save the rest of the reference information for when you need it.

2.3 Connectors and Indicators

Follow these steps to make the initial connection to the ValuPoint VP6-1460.

- (a) Connect power. Apply +12 to +24VDC or 24VAC to the terminal marked "POWER",

and common or ground the terminal marked "GND". (DC power is recommended.)



(b) Connect a CAT5 cable between the RJ-45 jack on the ValuPoint, and your network switch or hub. You cannot connect directly to your PC unless you use a "crossover" cable.

(c) Apply power.

A blue LED inside the case should light indicating power is present.

If the link LED on the RJ45 jack is not on, check your Ethernet cable connections. Both link and activity LEDs on the RJ45 jack will be on solid for a short time during boot-up. The entire bootup process will take about 20 seconds, during which time you will not be able to connect with a browser.

Ethernet link LED is the yellow LED integrated into the CAT5 connector. Ethernet activity LED is the green LED integrated into the CAT5 connector.

Refer to Appendix A for additional detail pertaining to connections and indicators as well as optional internal jumper settings.

2.4 Web User Interface

The default IP address as shipped is 10.0.0.101. Enter `http://10.0.0.101` in your browser's address window. Newer computers should be able to connect directly to that IP address. Older computers required that the PC be on the same subnet first, or that you add a route to your network configuration.

This generally works, but if this fails, you will need to temporarily change your computer's IP address to a fixed address that starts with 10.0.0. and ends with anything but 101.



Open your browser, and enter "http://10.0.0.101/" in the address window. You should see a page with the "ValuPoint IoT VP6-1460" header shown above. From this point, you will find help on each page in the web site contained within the product.

When you click on any of the page tabs such as System, you will be asked for a user name and password. The only default login as shipped is "root". The password is different for every ValuPoint shipped, and unique to your ValuPoint. Look for the root password document and/or label that was shipped with your device. If you have lost your root password, you will need to open a support ticket at <https://ticket.csimmn.com> and provide the MAC address shown so that your original default password can be recovered. Or you can follow the procedure described in Appendix section A.6.

To change the IP address of the ValuPoint, go to the Network page under System :: System Setup. The following page should appear (only top portion illustrated here). Change the IP address, and subnet mask and gateway if applicable. Click Change IP to save the changes. The process of programming this into Flash takes around half a minute. The new IP address only takes effect following the next system restart or power cycle.

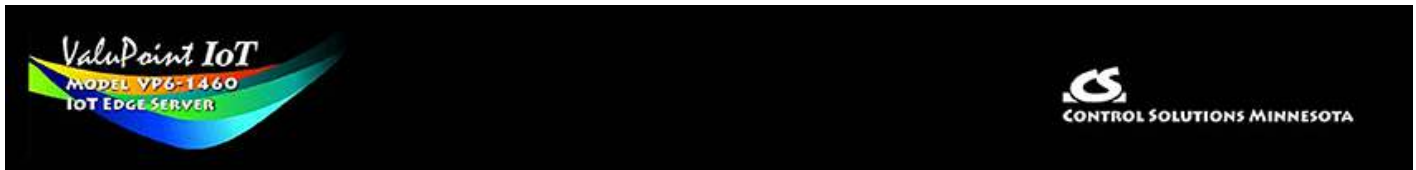
The screenshot shows the 'Network' configuration page in the ValuPoint web interface. The page is organized into several sections:

- IPv4 Settings:** Radio buttons for 'Automatic' and 'Static' (selected). Fields for IPv4 Static IP Address (192.168.1.183), IPv4 Static Subnet Mask (255.255.255.0), and IPv4 Static Gateway (192.168.1.1). Corresponding 'Configured' and 'Subnet Mask' fields are also present, along with an 'Apply' button.
- IPv6 Settings:** Radio buttons for 'Disabled' (selected), 'Automatic', and 'Static'. Fields for IPv6 Link-Local IP Address (fe80::240:9dff:fedc:ddd), IPv6 Configured IP Address (---), IPv6 Static IP Address (---), IPv6 Prefix Length (64), and IPv6 Gateway Tunnel (::).
- DNS Settings:** Fields for Primary DNS (1.1.1.1) and Secondary DNS (8.8.8.8), with their respective hexadecimal representations (::FFFF:1.1.1.1 and ::FFFF:8.8.8.8).

Most changes are stored in an XML configuration file in the device's Flash file system. Only a few are stored differently, and the IP address is one of those. Normally, clicking Update on any configuration page only stores that configuration information to a temporary RAM copy of the configuration file. To make your changes other than IP address permanent, you must execute Save XML Config File on the File Manager page (System :: System Setup :: File Manager). Refer also to section 3.1.

2.5 External Programming Tool

The VP6-1460 has two programming environments. The Script Basic environment is accessible via the web user interface and is discussed in section 18 of this user guide. The graphical programming environment is independent of the web user interface and is covered in a separate user guide supplement. The graphical programming environment is available for all VP6-14xx models and will function the same for both web and non-web versions of ValuPoint.



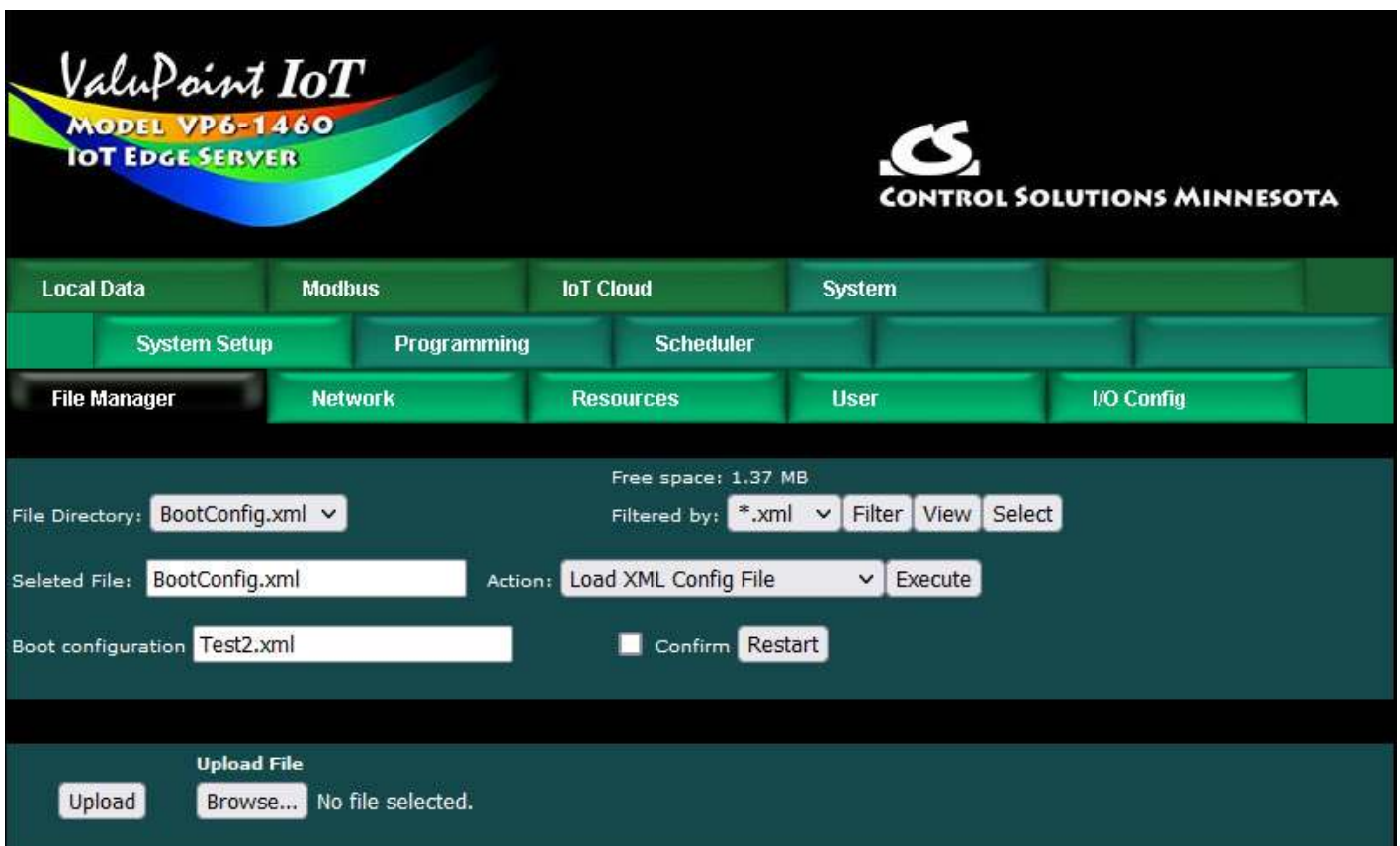
3. System Configuration and Resources

3.1 File Manager

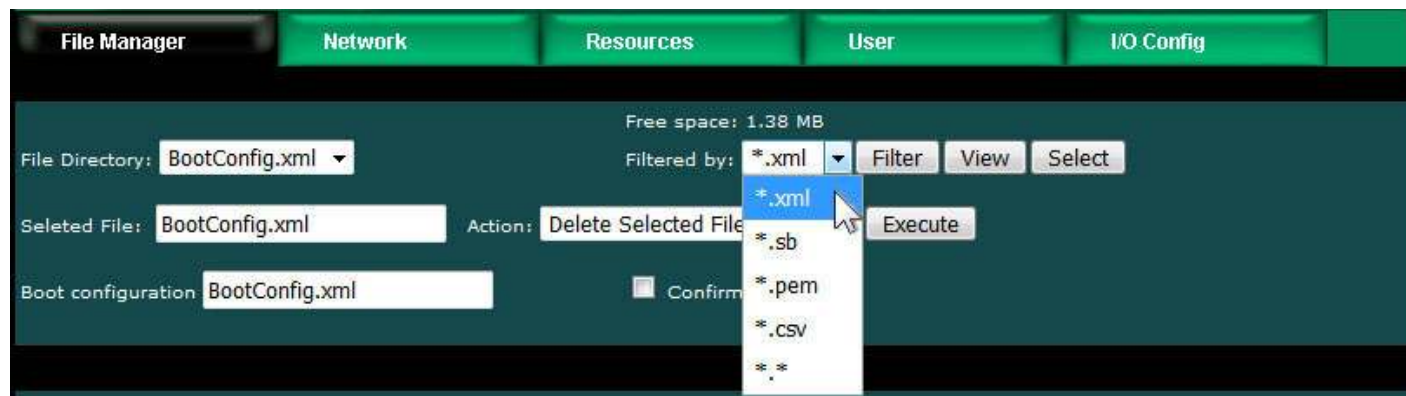
The File Manager page is probably one of the most important pages to know about. Among other things, this is where you tell the ValuPoint to save all of the changes you have made. The various "Update" buttons on the many pages in the web user interface only copy your configuration from your PC's browser to temporary memory in the ValuPoint. To retain those changes indefinitely (i.e. through restart or power cycle), you need to tell the ValuPoint to save those changes in a configuration file.

The configuration files are stored in non-volatile (Flash) memory. The process of reprogramming the Flash takes a little time. It would be cumbersome to rewrite that file every time you made a minor change. Therefore, in the interest of being more responsive, and in the interest of extending the life of the Flash, configuration is only saved to Flash when you direct it to do so.

The File Manager is used in several other ways in addition to managing your XML configuration files. You upload Script Basic programs here. You upload SSL certificates here. You import CSV files for Modbus configuration here.



The File Directory is a list of files that are currently stored in the ValuPoint's Flash file system. To filter files by type, select a type from the Filtered by list, and click Filter.

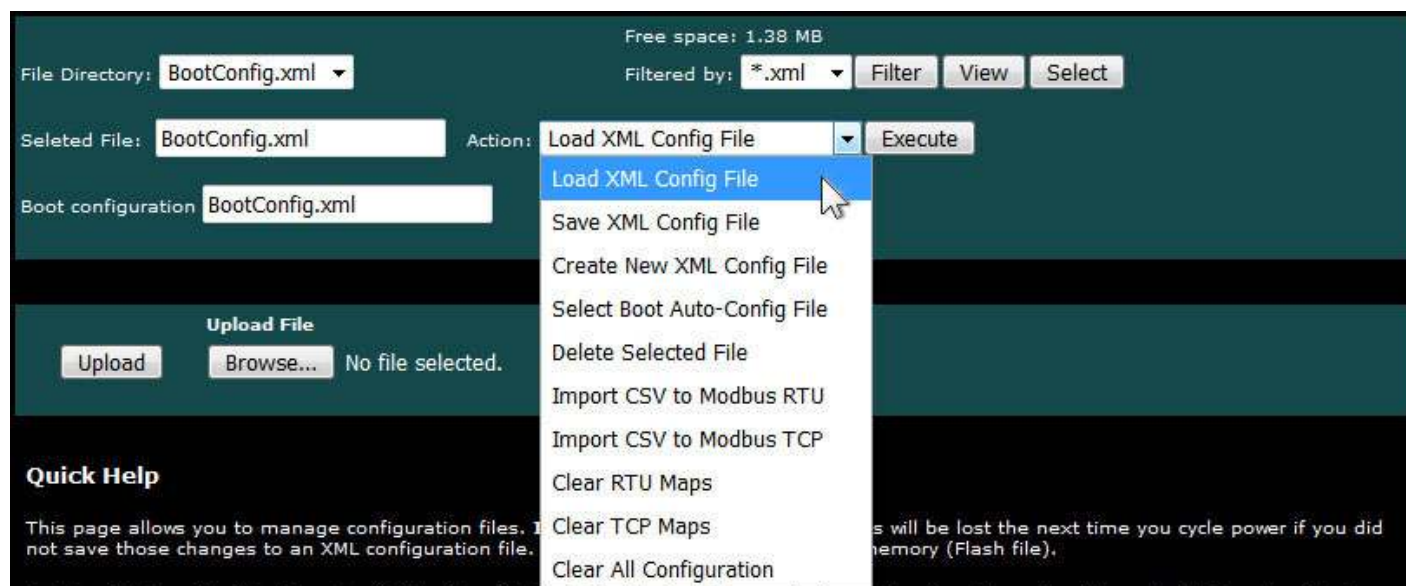


File type filters are as follows:

- *.xml XML configuration files
- *.sb Script Basic programs
- *.pem SSL certificates (for AWS IoT and/or HTTPS)
- *.csv CSV spreadsheet for Modbus register import
- *.txt Text file used as email message template
- *.* Display all files

There are several file related actions you may take. To take action with a certain file, select that file from the File Directory list, and click Select. That file should now show up in the Selected File window.

Once a file has been selected, choose your action from the Action list, and click Execute.



You must use the Select button to populate the Selected File window prior to executing any action from the list. Choose a file from the drop down list that shows all available files, then click the Select button. You may then act on that file.

You do not need to use the Select button to simply View a file. Clicking View will cause your browser to display the file chosen from the drop down list. If you attempt to View a CSV file, your PC will likely ask if you want to download the file or open it with your spread sheet program (e.g. Excel).

Upload File: To upload a file from your PC to this ValuPoint, use the Browse button to find the file on your PC, open the file in the PC's file dialog box, and then click Upload.

NOTE: If you get a message about directory needing synchronizing, click the browser's "back" button again to return to this page and click Upload again. This gets the browser and HTTP server back in sync, and this requirement generally happens only once or twice following power-up.

Restart: To restart the ValuPoint, check Confirm and click Restart. This is a hard reset that will accomplish the same thing as a power cycle without physically disconnecting and reconnecting power.

3.1.1 Load, Save, Create XML Configuration File

Load XML Config File: The configuration file shown in the "Boot configuration" window will be loaded automatically at startup. If you have uploaded a new configuration file and wish to use it without restarting, select that file and select this action.

HINT: If you are loading a file generated externally and you get "parameter out of range" errors pertaining to defining registers or "table full" errors while loading maps or rules, you might not have sufficient resources allocated. You may need to increase some counts on the Resources page.

Save XML Config File: Any time you have made configuration changes that you want to retain as permanent, you need to come here, select the file from the directory list, and execute this Save action.

Create New XML Config File: You have the option to a totally new configuration file. This is often suitable if you started with an existing configuration, made changes, and want to save your changes without replacing the original configuration. To create a new file, rather than selecting a file from the directory list, simply type a new name into the Selected file window. The name cannot contain spaces or special characters, and be sure to use the correct file suffix. Enter the name and execute this action.

3.1.2 Select Startup Configuration

Select Boot Auto-Config File: This is where you tell the ValuPoint what configuration to automatically load upon startup. To set the Boot configuration, select the XML file from the list, and execute this action. The name of the startup file, along with a few other important things like the ValuPoint's own IP address, are stored in a different area of Flash that is not part of the file system.

When selecting a new Boot configuration file, it is a good idea to select the file, and execute Load XML Config File. If there are errors, they will be displayed. If there are

errors in the file but you do not fix them, then the ValuPoint will not fully start up the next time it restarts. The web user interface will be available, but it will not be talking to Modbus devices.

3.1.3 Delete a File

Delete Selected File: Remove a file from the Flash file system by selecting it and executing this action.

3.1.4 Import CSV File

Import CSV to Modbus RTU: You can configure Modbus RTU read and write maps in bulk by importing the maps as a CSV file that you created using a standard spreadsheet program. Refer to Appendix B for details about the CSV format. Note that maps will be added to the existing map list. If you want to replace existing maps with imported maps, execute Clear RTU Maps first.

Import CSV to Modbus TCP: You can configure Modbus TCP read and write maps in bulk by importing the maps as a CSV file that you created using a standard spreadsheet program. Refer to Appendix B for details about the CSV format. Note that maps will be added to the existing map list. If you want to replace existing maps with imported maps, execute Clear TCP Maps first.

HINT: If you get "table full" errors while importing CSV files, you might not have sufficient resources allocated. You may need to increase some counts on the Resources page.

3.1.5 Clear Configuration

Clear RTU Maps: Execute this action to clear (completely remove) all Modbus RTU read and write maps.

Clear TCP Maps: Execute this action to clear (completely remove) all Modbus TCP read and write maps. The Modbus TCP device table will be left intact.

Clear All Configuration: Execute this action to completely wipe out all configuration. This includes all Modbus maps and devices, all IoT configuration, and all local registers. This will put you back to a "reset to factory" condition with the exception that your IP address is left unchanged. (See Appendix A, Section A.6, regarding forced hard configuration reset that includes IP address and root password.) If you want to make the now empty configuration permanent, select the file that is also selected as Boot configuration, and execute the Save XML Config File action.

The other means of completely wiping out all saved configuration is to simply delete the file named as the Boot configuration file, and then restart or power cycle the ValuPoint. Upon restart, a new empty configuration file will be created automatically.

3.2 Configuration Files and Restoring Default Settings

There is a means of restoring the ValuPoint to "manufacturer's default settings". First of all, make sure that the Boot configuration file is set to "BootConfig.xml". Then, after selecting this file as the boot file, delete it. Now restart the ValuPoint. Upon restart, and upon finding that the boot configuration name is BootConfig.xml, and it does not exist, the ValuPoint will automatically create one with default parameters. The automatic creation of a default file will not occur with any other file name.

Manual Editing: It is possible to manually edit the XML file outside of the ValuPoint. However, doing so is very prone to errors. If there are errors in the XML file, it will not load successfully on startup. If the configuration does not load on startup, none of the scanners will begin scanning. Because they are all blocked by configuration failure, entering new configuration via the web pages will not result in functionality being restored. You must successfully load a configuration file before the ValuPoint will become functional. To check for errors, select the file here, select Load XML Config File, and click Execute. Error messages that would have been discarded by the automatic loading at startup will now be displayed on an error page if there are any.

Backup Copy of XML Config File: To save a copy of the configuration to your PC, select the file and click the View button. Your browser will now display the XML file. DO NOT do a text copy/paste to try to create an XML file - doing so will result in an invalid file format that cannot be loaded again. You must use the browser's "save as" or "save page" function. The browser should default to wanting to save a file with a .xml suffix. If correctly saved on your PC, you should be able to double click on the saved file and it will result in opening the file automatically in your browser. It was saved correctly if the browser does not give any error messages when displaying the XML (which should now look exactly as it did when you first clicked the View button). Saving the configuration file to your PC, and then uploading on a different device, is a quick and easy way to configure two ValuPoints the same way.

Note about caching: Your browser may cache files. If you view a file, make configuration changes, save the file, then view the file again, you may see the old file cached by the browser. To see the updated file, go to "Options" in your browser's tools menu, and delete temporary Internet files (or delete cache files). Also, if you upload a file, make changes on your PC, and re-upload the same file, the browser may send the old file. Again, you will need to find the button inside your browser options that lets you delete the cached files from your PC. To upload a configuration file from your PC to the ValuPoint, use the Browse button to find the file on your PC, open the file in the PC's file dialog box, and then click Upload.

3.3 Network Configuration

The Network Configuration page is where you set the ValuPoint's IP address as well as a few other important things.

Local Data	Modbus	IoT Cloud	System
System Setup	Programming	Scheduler	
File Manager	Network	Resources	User
			I/O Config

IPv4 Settings Automatic Static

IPv4 Static IP Address IPv4 Configured IP Address **192.168.1.120**

IPv4 Static Subnet Mask IPv4 Subnet Mask **255.255.255.0**

IPv4 Static Gateway IPv4 Gateway **192.168.1.1**

IPv6 Settings Disabled Automatic Static

IPv6 Link-Local IP Address **fe80::240:9dff:fe45:4710**

IPv6 Configured IP Address **fec0::d**

IPv6 Static IP Address

IPv6 Prefix Length

IPv6 Gateway Tunnel

DNS Settings

Primary DNS ::FFFF:1.1.1.1

Secondary DNS ::FFFF:8.8.8.8

3.3.1 IPv4, IPv6 Settings

To change the IP address(es) of this device, make the applicable entries and click Apply. The "automatic" selection means DHCP. Changes to the IPv4 IP address will take effect upon the next system restart.

If IPv6 is enabled, IPv6 will always have a Link-Local address, plus one configured address. The configured address will be either the static IP address, or an IPv6 address obtained from an IPv6 DHCP server. If no configured address appears, the DHCP server may have been unreachable.

The IPv6 static IP address window is the configured static address. If "Static" is selected and a new IP address entered as the static address, this new address will not take effect until the next system restart.

The numbers shown to the right of the IPv4 input windows are the actual numbers currently in use. If static IP addresses have been entered but the ValuPoint has not been restarted yet, these numbers will not be the same.

You may use domain names instead of static IP addresses in several instances. If domain names are used, you must supply the IP address of at least one DNS server here. The DNS server must be at a static IP address. These changes take effect immediately. Note: If you are using DHCP, the DNS addresses will be supplied by the DHCP server and should be set to 0.0.0.0 here.

3.3.2 NTP Time Server Settings

The ValuPoint maintains time and date via SNTP services or its internal

Real Time Clock/Calendar (RTC). The RTC can also be used as backup should SNTP be unavailable due to network disconnect.

The screenshot shows a configuration window with the following fields and values:

- Primary NTP Server: 132.163.96.2
- Secondary NTP Server: 129.6.15.30
- Daylight Time Start Rule: 3.2.0/02:00:00
- Daylight Time End Rule: 11.1.0/02:00:00
- Standard GMT Offset: -360 Minutes
- Daylight GMT Offset: -300 Minutes
- NTP Refresh Period: 300 Minutes
- Latitude: 45.062126
- Longitude: -92.984154
- Current Local Time: 2024-02-26 10:06:17
- Use RTC:
- Sunrise: 06:56
- Sunset: 17:55

NTP setup: Enter a primary and secondary IP address of NTP servers, such as those found at www.nist.gov (go to <http://tf.nist.gov/tf-cgi/servers.cgi> to find more). Enter daylight start/end rules, and offset from GMT for both standard and daylight time. Offset is a negative number in the western hemisphere. Enter an NTP update time in minutes. Do not set NTP to update too frequently or you risk being denied service by the NTP server. Click the Set NTP button after all settings have been made. The Flash update will take several seconds. The initial update of local time may take a minute or two. You may need to restart the ValuPoint if NTP had never before been initialized.

Daylight savings time start/end rules consist of "date/time" where the date (m.n.d) indicates the day when summer time starts or ends, and time (hour:min:sec) is the current local time when summer time starts/ends. The date portion of the rule is formatted as follows:

m indicates the month ($1 \leq m \leq 12$)

n indicates which week of the month ($1 \leq n \leq 5$). 5 = the last week in the month.

d indicates what day of the week ($0 \leq d \leq 6$). 0 = Sunday

For example: Start "4.1.0/02:00:00", end "10.5.0/02:00:00" means summer time starts at 2am on the first Sunday in April and ends at 2am on last Sunday in October. That was the old US rule. The new US rule is start "3.2.0/02:00:00" and end "11.1.0/02:00:00", which is start at 2am on the second Sunday in March, end at 2am on the first Sunday in November.

Latitude and longitude for the location of this device should be entered if you want to use the astronomical clock feature of the scheduler. Without latitude and longitude, the calculations for sunrise and sunset will be incorrect.

RTC (Real Time Clock) Setup: Check the "Use RTC" box, and enter the current date and time in the window below that box. Then click Set RTC. In order to use the scheduler without any SNTP, you do still need to enter Latitude and Longitude and click Set NTP. The RTC does not automatically adjust for daylight savings. If both NTP and RTC are enabled, then the RTC time/date will be updated from SNTP when available.

CAUTION: The lithium battery contained in this device may explode if mistreated. DO NOT recharge, disassemble, or dispose of in fire.

No action is required of the user to activate the battery that backs up the real time clock. Important: Replace battery with BR1225A only. Use of another battery may present a risk of fire or explosion.

3.3.3 Port Settings

Web Server HTTPS Enabled (on 443) HTTP Enabled

HTTP Port (default 80)

Modbus Port (default 502)

FTP Server Enabled

REST API Enabled

MAC Address: 00:40:9D:45:47:10 System Uptime: 3,10:27:41

HTTPS certificate status: Using self-generated X.509

Secure browsing can be enabled here, and non-secure can be disabled. You cannot disable both, and a forced configuration reset will restore HTTP (non-secure) web browsing. In order to use HTTPS, you must first upload the necessary SSL certificates (see Appendix D) or allow the certificates to be self-generated by explicitly deleting existing certificates.

IMPORTANT: It is highly recommended that in making the transition from HTTP to HTTPS, you enable both until you confirm HTTPS is functional. If there is a problem with the SSL certificates provided for HTTPS, then HTTPS will not run and you will find an error message on the "HTTPS certificate status" line. If you disable standard HTTP without first verifying that HTTPS is functional, you may end up locked out and will then need to do a forced hard reset (Appendix A.6).

The HTTP port for browsing the user interface can be moved away from the default HTTP port 80. Select a different port, click Set Ports, and then restart the ValuPoint to make that new port take effect. Don't forget to append the port number to the ValuPoint's IP address when attempting to browse the web user interface if it has been moved from port 80.

The Modbus port will be set to zero, meaning Modbus TCP is disabled, when the device is new. Enter the standard port 502 and click Set Ports to set the Modbus port. Set the port to some other port if you know that Modbus TCP operates on a non-standard port on your network. The device needs to be restarted after changing the Modbus TCP port.

FTP is enabled by default to allow firmware update uploads. It may be optionally disabled here. Just remember to enable it again before attempting a firmware update.

A REST API is available if you wish to query the ValuPoint and get replies to HTTP GET/POST requests in JSON format. The API will be disabled by default, but you enable it here if desired. Refer to Section 21 for details about the API.

Any changes to this port numbers or enabling/disabling features requires restarting the ValuPoint before they will take effect.

3.4 Resource Allocation

Historically, Control Solutions devices had a fixed set of resources to work with. Invariably, there were always users that wanted less of this and more of that. Therefore, while there are still maximums imposed, you can now shift resources around as best suits your application. An example is shown below.

The values in the Pending column are those found in the most recently loaded XML configuration file. When saving or creating a new XML file, the numbers in the Current column will be written to the file. To change the allocations, change numbers in the Pending column. When you are ready to commit these changes, click the Commit button. To cause the changes to go into use, you must restart the device since memory allocation can occur only once at startup.

You can click the Check button prior to Commit to see if the values you have entered will be accepted. If adjustments need to be made, the values in the Pending column will be updated.

The first time you visit this page, you will see the initial default values. Should you change any of them, minimums and maximums currently defined in firmware will be imposed. If you see a value smaller than what you entered, it may be that you had exceeded the internal limit.

If you see that numbers toward the top of the list are large, and numbers near the bottom are all set to 1, it means the system has run out of free memory and you need to reallocate resources.

Resource	Current	Pending
Local Modbus Register Pool Size	400	<input type="text" value="400"/>
Number of Data Calculate Rules	100	<input type="text" value="100"/>
Number of Data Copy Rules	100	<input type="text" value="100"/>
Number of Event Notify Rules	100	<input type="text" value="100"/>
Number of Scheduler Weekly Events	50	<input type="text" value="50"/>
Number of Scheduler On Demand Events	20	<input type="text" value="20"/>
Number of Modbus RTU Read Maps	200	<input type="text" value="200"/>
Number of Modbus RTU Write Maps	50	<input type="text" value="50"/>
Number of Modbus TCP Devices	10	<input type="text" value="10"/>
Number of Modbus TCP Client Read Maps	100	<input type="text" value="100"/>
Number of Modbus TCP Client Write Maps	20	<input type="text" value="20"/>
Number of Modbus TCP Server Connections	20	<input type="text" value="20"/>
Estimated Memory Utilization	5.23%	5.23%
Persistent Data Status	Ok	<input type="button" value="Reset Persistent Data"/>

The estimated memory utilization shown at the bottom gives you an indication of how close you are to running out of memory. You will not be allowed to commit a resource allocation greater than 100%.

There is no magic formula for determining the "right" size configuration. All resources are allocated from "free memory". This free memory is also used by Script Basic for its variables. Therefore, allocating fewer resources for things on this list will make more memory available for Script Basic.

The Reset Persistent Data button is used to clear (zero) stored persistent data. Changing the local register pool size will clear persistent data. To clear data without changing any allocations, use this button. Local registers have the option of being configured as "persistent" which means the value contained in that register will be retained through restart or power cycle.

3.5 User Login Passwords

There is only one default login provided initially. That login is the username "root" and root's password is a unique password generated specifically for this particular ValuPoint. That unique password was provided for you in documentation included with the shipment. That unique password complies with California Consumer Privacy Act SB-327, which requires all Internet connected devices to have unique default passwords.

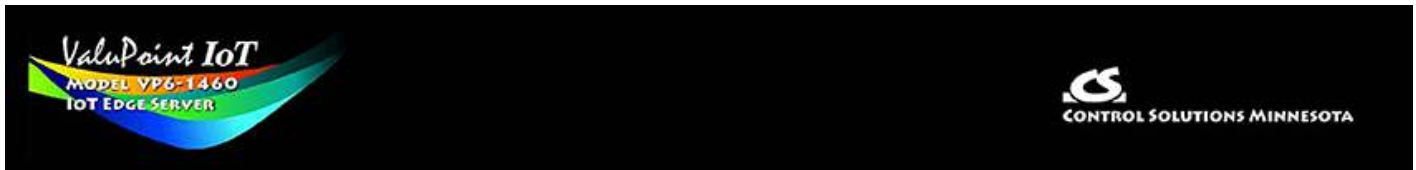
Once logged in as "root", you have the option of creating up to five additional logins.

The privilege level Administrator lets that user see and change anything. The privilege level Maintenance allows the user to log in and see (and change) values in the local registers via the Local Registers page, but cannot access any other pages. The Restricted level will block access to everything except user defined web pages.

You also have the option of IP filtering. If set, then the user can only access ValuPoint's web pages from that IP address. Leave set to 0.0.0.0 to disable filtering.

Only the root user will see this version of the User page. Other users will only be able to change their own password. To add or change a user, enter the name and credentials, check Confirm Change, and click Change. To delete a user, clear the name field, check Confirm Change, and click the Change button.

Local Data		Modbus		IoT Cloud		System	
System Setup			Programming		Scheduler		
File Manager		Network		Resources		User	
						I/O Config	
							Change
User Name	Password	Privilege Level	IP Filter	Confirm Change			
system	•••••	Administrator ▼	0.0.0.0	<input type="checkbox"/>			
		Restricted ▼	0.0.0.0	<input type="checkbox"/>			
		Restricted ▼	0.0.0.0	<input type="checkbox"/>			
		Restricted ▼	0.0.0.0	<input type="checkbox"/>			
		Restricted ▼	0.0.0.0	<input type="checkbox"/>			
root	••••••	Unrestricted	0.0.0.0	<input type="checkbox"/>			
root confirm							



4. Configuring Local Registers

4.1 Creating Local Registers

Control Solutions devices originally came with a predefined set of local registers that were accessible externally as Modbus registers. Newer models take a completely different approach to defining registers. When you first take the ValuPoint out of the box, it has no registers except for the initial set of registers assigned to the physical I/O points. You get to create remaining registers as you need them, and several data formats are supported, from 16-bit to 64-bit, both integer and floating point.


Your first visit to the Local Registers page will be as illustrated below.

Local Register #	Register Name	Set	Register Data	Register Format
00001	A/UI #1	<input type="checkbox"/>	0.000000	Single Float
00003	A/UI #2	<input type="checkbox"/>	0.005569	Single Float
00005	A/UI #3	<input type="checkbox"/>	0.000000	Single Float
00007	A/UI #4	<input type="checkbox"/>	0.005569	Single Float
00009	A/UI #5	<input type="checkbox"/>	0.000000	Single Float
00011	A/UI #6	<input type="checkbox"/>	0.005569	Single Float
00013	A/UI #7	<input type="checkbox"/>	0.000000	Single Float
00015	A/UI #8	<input type="checkbox"/>	0.005569	Single Float
00017	A/UI #9	<input type="checkbox"/>	0.000000	Single Float
00019	A/UI #10	<input type="checkbox"/>	0.002785	Single Float
00021	A/UI #11	<input type="checkbox"/>	0.008354	Single Float
00023	A/UI #12	<input type="checkbox"/>	0.002785	Single Float
00025	DO #1	<input type="checkbox"/>	0	Unsigned 16-bit
00026	DO #2	<input type="checkbox"/>	0	Unsigned 16-bit

You do not need to create any additional registers if you will only be using the I/O

points. To begin the process of creating additional registers, click on the last register number on the list. Later on, click on the register number at the end of the list to add more, or click anywhere in the middle of the list if there are gaps in the register number sequence that you would like to fill.

00023	A/UI #12	<input type="checkbox"/>	0.002785	Single Float
00025	DO #1	<input type="checkbox"/>	0	Unsigned 16-bit
00026	DO #2	<input type="checkbox"/>	0	Unsigned 16-bit

 Quick Help

Upon clicking a register number, the register detail will be displayed. This can be either detailed configuration of an existing register, or detail about new registers you are about to add. The only time you can select the data format is when adding new registers. Once the registers are created, the data format cannot be changed because this impacts how many Modbus registers are actually used. If you had a set of 16-bit registers defined, and wanted to change one of them to 32-bit, it would cause all of the remaining registers to be renumbered if such a change was allowed. While this may seem harmless at first, it becomes a huge mess trying to keep track of where in all the rules the existing numbers needed to get changed. Therefore, such changes are prohibited.

Select the data format for the new registers to be created. The designations Signed and Unsigned refer to integers. Float refers to IEEE 754 floating point format. Character string refers to a series of registers with two ASCII characters per 16-bit register. Mod10 refers to a format unique to Schneider Electric power meters (refer to Schneider Electric documentation for a definition of those formats).

IMPORTANT: Modbus protocol knows holding registers (or input registers) to simply be a 16-bit piece of data. The protocol knows nothing about signed or unsigned integer - that interpretation is up to you. The 16 bits may even be a collection of 16 status flags. If a register is defined in the ValuPoint as anything bigger than 16 bits, it is actually a pair (or series of 2 or more) of 16-bit registers. Again, Modbus protocol does not know anything about floating point, character strings, etc. Modbus only knows how to send 16 bits of something at a time when function codes reference a holding register or input register. Modbus only knows how to send 1 bit at a time if referenced as a coil or discrete input. It is up to the Modbus master to be smart enough to ask for 2 registers at a time if it knows it wants to read a 32-bit value.

The screenshot shows a software interface for configuring local registers. At the top, there are tabs for 'Local Registers', 'Calculate', 'Copy', and 'Report'. Below these, there are input fields for 'Register #' (26) and 'Links: DO 2---', along with 'Update', '< Prev', and 'Next >' buttons. The main configuration area includes a 'Register data format' dropdown menu (currently set to 'Unsigned 16-bit'), a 'Size' field (0), and a 'Register name' field ('Data Value 1'). There are also checkboxes for 'Least significant data', 'Display as hexadecimal', and 'Is Persistent'. A 'Quick Help' section is visible in the bottom left corner.

In addition to selecting data format for creating new registers, provide a temporary register name. You can change this later, but it is usually helpful to start with something for a name, and it is suggested that the name ends with a number. The reason why is illustrated shortly.

Modbus protocol is strict about 16-bit increments of data for holding registers. However, when register pairs (or quads) are used to hold a 32-bit (or 64-bit) value, the order in which those registers are interpreted is not defined by any standard. It is up to you to keep track of that. ValuPoint supports interoperability with other Modbus devices by letting you specify what order should be used internally. If the least significant data should appear in the first (or lower numbered) register, then check the box that says "Least significant data should be in first register" either when creating the register, or later by reconfiguring the existing register.

When first creating registers, you do not need to enter any of the default information on the last line. (You can, but don't have to.) The size only applies when creating character string variables (illustrated later, below).

The first register number to add, indicated at the bottom of the page, will be the first available register slot that is not yet assigned. You can enter some different number here. It is not required to create registers in contiguous order. You can jump around, as long as registers don't try to overlap. Select a number of registers to create, and click Add New. If you attempt to add registers that overlap existing registers, the allocation algorithm will find an available slot for you and fill that instead. The maximum possible register number will be the count indicated on the Resources page as Local Modbus Register Pool Size.

We have now created 10 new floating point registers. The important thing to observe here is the register numbers in the first column. Remember that Modbus protocol assigns register numbers (or addresses) in 16-bit increments. Since single precision floating point registers are 32-bit registers, each floating point value occupies 2 spots in the Modbus register map. The local register number assignments reflect this fact.

Note that the name given in the screen above was "Data Value 1" but our series of 10 new registers came up with 10 sequential names. If the last thing to appear in the name given when assigning new registers is a number, this value will be incremented by one in the name of each successive new register.

Local Register #	Register Name	Set	Register Data	Register Format
00025	DO #1	<input checked="" type="checkbox"/>	0	Unsigned 16-bit
00026	DO #2	<input checked="" type="checkbox"/>	0	Unsigned 16-bit
00027	Data Value 1	<input checked="" type="checkbox"/>	0.000000	Single Float
00029	Data Value 2	<input checked="" type="checkbox"/>	0.000000	Single Float
00031	Data Value 3	<input checked="" type="checkbox"/>	0.000000	Single Float
00033	Data Value 4	<input checked="" type="checkbox"/>	0.000000	Single Float
00035	Data Value 5	<input checked="" type="checkbox"/>	0.000000	Single Float
00037	Data Value 6	<input checked="" type="checkbox"/>	0.000000	Single Float
00039	Data Value 7	<input checked="" type="checkbox"/>	0.000000	Single Float
00041	Data Value 8	<input checked="" type="checkbox"/>	0.000000	Single Float
00043	Data Value 9	<input checked="" type="checkbox"/>	0.000000	Single Float
00045	Data Value 10	<input checked="" type="checkbox"/>	0.000000	Single Float

Now let's proceed to add some character strings. Click on the last register number assigned thus far to open the register detail page.

00041	Data Value 8	<input type="checkbox"/>	0.000000	Single Float
00043	Data Value 9	<input type="checkbox"/>	0.000000	Single Float
00045	Data Value 10	<input type="checkbox"/>	0.000000	Single Float

Quick Help

Select Character String for data format. This is the one time a size must be specified. In the example below, we are going to allocate 40-character strings. This means that each "register" will actually be a series of 20 Modbus registers (two ASCII characters per register). The character string processing assumes that any display device used in conjunction with these registers will display the high order byte on the left and low order byte on the right in any given 2-character portion of the display screen. This is consistent with display devices that have been tested with ValuPoint.

Local Registers Calculate Copy Report

Register # Links: ----- Update < Prev Next >

Register data format: Size: Register name

Least significant data should be in first register: Display as hexadecimal: Is Persistent:

Apply this default value: At power-up If not updated by remote source within seconds.

First register number to add: Add this many: **Add New** Register # Delete

After clicking Add New, we now see our character string registers in our register list (click on the Local Registers tab to get back to the register list). Note that the register numbers increment by 20 to accommodate our 40-character strings.

Local Registers Calculate Copy Report

Showing registers from Update < Prev Next >

Local Register #	Register Name	Set	Register Data	Register Format
00027	Data Value 1	<input type="checkbox"/>	0.000000	Single Float
00029	Data Value 2	<input type="checkbox"/>	0.000000	Single Float
00031	Data Value 3	<input type="checkbox"/>	0.000000	Single Float
00033	Data Value 4	<input type="checkbox"/>	0.000000	Single Float
00035	Data Value 5	<input type="checkbox"/>	0.000000	Single Float
00037	Data Value 6	<input type="checkbox"/>	0.000000	Single Float
00039	Data Value 7	<input type="checkbox"/>	0.000000	Single Float
00041	Data Value 8	<input type="checkbox"/>	0.000000	Single Float
00043	Data Value 9	<input type="checkbox"/>	0.000000	Single Float
00045	Data Value 10	<input type="checkbox"/>	0.000000	Single Float
00047	Char String 1	<input checked="" type="checkbox"/>	Hello Modbus	Char String[40]
00067	Char String 2	<input checked="" type="checkbox"/>	String can be read by Modbus	Char String[40]
00087	Char String 3	<input checked="" type="checkbox"/>	Or written by Modbus	Char String[40]
00107	Char String 4	<input type="checkbox"/>		Char String[40]

You can set the character strings from the web page,. The strings can be read or written by Modbus as a series of registers with 2 characters per register.

4.2 Configuring Physical I/O

The first 26 registers in the ValuPoint are permanently assigned to the physical I/O points. Those register numbers are illustrated in the first screen shot in Section 4.1 above.

ValuPoint IoT
MODEL VP6-1460
IOT EDGE SERVER

CONTROL SOLUTIONS MINNESOTA

Local Data | Modbus | IoT Cloud | System

System Setup | Programming | Scheduler

File Manager | Network | Resources | User | **I/O Config**

Update

A/UI #	Hardware Type	Scale	Offset	Qualifier	Is Persistent
1	0-10V, 12-bit	0.00	0.00	0	<input type="checkbox"/>
2	0-10V, 12-bit	0.00	0.00	0	<input type="checkbox"/>
3	4-20mA, with resistor	6.250000	-25.00000	499	<input type="checkbox"/>
4	4-20mA, with resistor	6.250000	-25.00000	499	<input type="checkbox"/>
5	Pulse counter	0.00	0.00	0	<input checked="" type="checkbox"/>
6	Pulse counter	0.00	0.00	0	<input checked="" type="checkbox"/>
7	Pulse counter	0.00	0.00	0	<input checked="" type="checkbox"/>
8	Pulse counter	0.00	0.00	0	<input checked="" type="checkbox"/>
9	Dry contact, active closed	0.00	0.00	50	<input type="checkbox"/>
10	Dry contact, active closed	0.00	0.00	50	<input type="checkbox"/>
11	Dry contact, active closed	0.00	0.00	50	<input type="checkbox"/>
12	Dry contact, active closed	0.00	0.00	50	<input type="checkbox"/>

Coprocessor Status: Ok (v6.01.8)

The relay outputs do not require any configuration. The analog/universal inputs do require configuration on the I/O Config page. Select the desired input type and enter additional parameters as applicable.

File Manager		Network		Resources		User		I/O Config	
Update									
A/UI #	Hardware Type			Scale	Offset	Qualifier	Is Persistent		
1	0-10V, 12-bit			0.00	0.00	0	<input type="checkbox"/>		
2	Unconfigured			0.00	0.00	0	<input type="checkbox"/>		
3	0-10V, 12-bit			6.250000	-25.00000	499	<input type="checkbox"/>		
4	4-20mA, with resistor			6.250000	-25.00000	499	<input type="checkbox"/>		
5	Discrete, active high			0.00	0.00	0	<input checked="" type="checkbox"/>		
6	Discrete, active low			0.00	0.00	0	<input checked="" type="checkbox"/>		
7	Dry contact, active open			0.00	0.00	0	<input checked="" type="checkbox"/>		
8	Dry contact, active closed			0.00	0.00	0	<input checked="" type="checkbox"/>		
9	Pulse counter			0.00	0.00	50	<input type="checkbox"/>		
10	Resistance			0.00	0.00	50	<input type="checkbox"/>		
11	Position pot, 1K-30K			0.00	0.00	50	<input type="checkbox"/>		
12	Thermistor, 10K type III, F			0.00	0.00	50	<input type="checkbox"/>		
	Thermistor, 10K type II, F			0.00	0.00	50	<input type="checkbox"/>		
	Thermistor, 3K type II, F			0.00	0.00	50	<input type="checkbox"/>		
	Thermistor, 20K type IV, F			0.00	0.00	50	<input type="checkbox"/>		
	Thermistor, 5K type II, F			0.00	0.00	50	<input type="checkbox"/>		
	Thermistor, 10K type III, C			0.00	0.00	50	<input type="checkbox"/>		
	Thermistor, 10K type II, C			0.00	0.00	50	<input type="checkbox"/>		
	Thermistor, 3K type II, C			0.00	0.00	50	<input type="checkbox"/>		
	Thermistor, 20K type IV, C			0.00	0.00	50	<input type="checkbox"/>		
	Thermistor, 5K type II, C			0.00	0.00	50	<input type="checkbox"/>		

The A/UI inputs may function as any of the following:

- 0-10V, 12-bit resolution (reading is Volts)
- 4-20mA, with resistor (reading is mA)
- Discrete, active high
- Discrete, active low
- Dry contact, active open
- Dry contact, active closed
- Pulse counter
- Resistance (reading is ohms)
- Position pot, 1K-30K (reading is percentage)

- Thermistor, 10K type III, F (readings in degrees for all of following)
- Thermistor, 10K type II, F
- Thermistor, 3K type II, F
- Thermistor, 20K type IV, F
- Thermistor, 5K type II, F
- Thermistor, 10K type III, C
- Thermistor, 10K type II, C
- Thermistor, 3K type II, C
- Thermistor, 20K type IV, C
- Thermistor, 5K type II, C

Dedicated hardware is available for pulse counting on channels 5, 6, 7, and 8. The only limiting factor on maximum pulse rate on these inputs is the noise filtering on the inputs. The inputs have been verified to count at up to 1kHz provided the signal amplitude is sufficient. Pulse counting is supported on the remaining input channels, but the counting is done by software and therefore the rate is limited to about 2Hz.

The 4-20mA setting is used as a label since that is most common. However, the input is actually going to be 0-20mA, and the register value for that range will be 0..20. The scale and offset can be used to convert 4-20mA to a 0-100% value by using scale = 6.25 and offset = -25. A dropping resistor of 500 ohms will use the full 10 volt input range. A dropping resistor of 250 ohms can be used - it will simply provide less resolution.

Scale – Scaling applies the formula $y=mx+b$. When reading from a hardware input, the raw data as read is multiplied by the scale factor, then the offset is added to produce the resulting register value. NOTE: If no scale factor is given (zero is entered), no scaling will be done, as if scale=1 and offset=0.

Offset – The offset portion of the scaling as noted above.

Qualifier – Enter the configuration qualifier value, if applicable, for the selected configuration. Qualifiers are required only in the following modes:

4-20mA mode: The qualifier is the resistance in ohms of the dropping resistor used to convert the current to voltage. An external resistor must be provided, connected between the A/UI input and ground/common. The resistor needs to be 1/2 watt (2 watt to withstand 24V power), and is left external simply because miswiring the 4-20mA sensor can easily apply 24V power directly to the input and cause the dropping resistor to heat up and possibly fail. The external resistor is simple to replace, whereas an internal resistor on a circuit board would be more trouble to replace.

Discrete and Dry Contact modes: The qualifier is a threshold between 1% and 99% at which the input should trip from off to on or vice versa. The A/UI inputs are specified as 0-10V inputs. Therefore, since discrete inputs are sampled as analog values and compared to a threshold, the qualifier here is a percentage of 10V for the trip point. A value of 50% will mean a threshold of around 5V.

Position Pot: Position is simply a different interpretation of resistance measurement. The value resulting from position pot measurement will be a percentage from 0% to 100%, but this percentage will be a ratio based on the resistance value in ohms provided as the qualifier.

4.3 Special Features of Local Registers

There are a couple of features that you may go back and change at any time after registers are created.

ValuPoint IoT
MODEL VP6-1460
IOT EDGE SERVER

CONTROL SOLUTIONS MINNESOTA

Local Data Modbus IoT Cloud System

Data Events

Local Registers Calculate Copy Report

Register # 101 Links: ----- Update < Prev Next >

Register data format: Unsigned 32-bit Size: 0 Register name Unsigned Value 1

Least significant data should be in first register: Display as hexadecimal: Is Persistent:

Apply this default value: 0.000000 At power-up If not updated by remote source within 45 seconds.

First register number to add: 127 Add this many: 1 Add New Register # 101 Delete

You may select hexadecimal display of data. This only applies to display on the local web pages, and does not change what Modbus sees externally (Modbus only sees a collection of bits, and that fact never changes). Hexadecimal display of a floating point value is probably meaningless, but hexadecimal display of registers containing a set of status bits packed into a single register is often easier to interpret when displayed as hexadecimal.

You have the option of applying a default value under two conditions: (1) Automatically at power-up; (2) Any time this register is not updated by some remote source within the timeout given (this assumes the ValuPoint is acting as a slave or server).

If this register is being used to hold a value provided by some other device acting as master, and you want a way of externally detecting when that device has failed to communicate, you can cause a "flag" value to appear in this register when a new value has not been received within the given timeout period.

Applying a default at power-up is potentially useful if you want this server to always write a certain value to some other Modbus slave any time the system wakes up.

The other use for default at power-up is when you need to use a constant value in a calculate rule. Set aside a register whose only purpose is to hold this constant for later use.

The "Is Persistent" option means this register will retain its most recent value through a power outage or restart, with time delay restrictions. ***The persistent selection does not take effect until the Reset Persistent Data button is clicked on the Resources page.***

The persistent data is stored in non-volatile memory which has a life expectancy of 4 million write cycles. However, if the data is written too frequently, the memory will still reach its end of life prematurely. Therefore, to extend the life of the EEPROM, persistent data is only updated once every 15 minutes. If the data has not been written in the past 15 minutes, new data will be written to EEPROM immediately. Otherwise writing to EEPROM will be delayed until the expiration of the timer for that register.

The most recently stored data is read from EEPROM and placed into the local register at startup.

4.4 Local Register Calculate Rules

The ValuPoint includes the ability to do simple calculations based on simple template rules. Select the operation, one or two operands as applicable, and a register to place the result in. Operations like "multiply" will use registers A "and" B. Operations like "sum" can add up the contents of a series of registers by selecting "thru" instead of "and".

These template rules can be useful for doing minor data manipulation or testing for purposes of enabling rules, or for generating MQTT Publish messages. If you need to do something more complex than what these simple rules will accomplish, you have the option of writing a program in Script Basic that can get as complicated as you like. Script Basic programs have the ability to both read and write local registers, which means your program can look at data values, and set registers that in turn cause things like MQTT messages to occur.

The screenshot shows the 'Calculate' configuration interface. At the top, there are tabs for 'Local Data', 'Modbus', 'IoT Cloud', and 'System'. Below these are sub-tabs for 'Data', 'Events', and 'Local Registers'. The 'Local Registers' sub-tab is active, showing a table with columns: Rule #, Perform Operation, Using Register #, And/Thru Using, This Reg#/Value, and Place Result in Register #. A dropdown menu is open for the 'Perform Operation' column of Rule #1, listing various operations. The 'Showing' box indicates '1 to 1 of 1' rules. Buttons for 'Update', '< Prev', and 'Next >' are visible. Below the table, there are 'Insert' and 'Delete' buttons and a '# Rules Enabled' field.

Here is an example of a template rule that multiplies register 27 by register 29 and places the result in register 31.

This screenshot shows the 'Calculate' configuration screen with two rules. Rule #1 is configured with 'Perform Operation' set to 'multiply', 'Using Register #' set to '27', 'And/Thru Using' set to 'and', 'This Reg#/Value' set to '29', and 'Place Result in Register #' set to '31'. Rule #2 is a placeholder with 'Perform Operation' set to 'none', 'Using Register #' set to '0', 'And/Thru Using' set to 'and', 'This Reg#/Value' set to '0', and 'Place Result in Register #' set to '0'. The 'Showing' box indicates '1 to 2 of 2' rules. The '# Rules Enabled' field shows '2'. Buttons for 'Update', '< Prev', 'Next >', 'Insert', and 'Delete' are present.

If registers 27 and 28 contain the values shown below, then the result shown in register 31 would be expected.

The screenshot shows the 'Local Registers' configuration screen. The 'Showing registers from' box is set to '27'. Below the navigation buttons, there is a table with columns: Local Register #, Register Name, Set, Register Data, and Register Format.

Local Register #	Register Name	Set	Register Data	Register Format
00027	Data Value 1	<input type="checkbox"/>	100.000000	Single Float
00029	Data Value 2	<input type="checkbox"/>	0.250000	Single Float
00031	Data Value 3	<input type="checkbox"/>	25.000000	Single Float

Constants may be introduced into the calculation by reserving a register to hold that constant, and then configuring it to apply a default value at power-up. This default

value should be the constant you wish to include in a calculation.

The other option is to use the set operation, but the value is limited to unsigned integer with the set operation. The example illustrated below will set register 33 to a value of 123.

Rule #	Perform Operation	Using Register #	And/Thru Using	This Reg#/Value	Place Result in Register #
1	set	33	using	123	33
2	none	0	and	0	0

Operations available on two or more registers using 'and' or 'thru':

add	Add two registers
subtract	Subtract second register from first
multiply	Multiply two registers
divide	Divide first register by second
sum	Sum two or more registers
average	Average two or more registers
min	Find lowest value among two or more registers
max	Find highest value among two or more registers
logic OR	Logically OR two or more registers
logic AND	Logically AND two or more registers
logic NOR	Logically NOR two or more registers
logic NAND	Logically NAND two or more registers
logic XOR	Logically Exclusive-OR two registers

Operations available on one register:

logic NOT	Generate bit-wise negation of register
test = 0	Set result to 'true' if register is zero
test < 0	Set result to 'true' if register is less than zero

test > 0	Set result to 'true' if register is greater than zero
----------	---

Operations available on one register 'using' a given value:

set	Set register to given value (unsigned 32-bit integer)
skip = N	Skip next operation if register is equal to given value
skip < N	Skip next operation if register is less than given value
skip > N	Skip next operation if register is greater than given value
comp = N	Compare, set result 'true' if register is equal to given value
comp < N	Compare, set result 'true' if register is less than given value
comp > N	Compare, set result 'true' if register is greater than given value
pack	Perform Pack operation (see text)
fill	Perform Fill operation (see text)
unpack	Perform Unpack operation (see text)
shift left	Shift content of register left by 'using' number of bits
shift right	Shift content of register right by 'using' number of bits

Operations "using" a given value will have an unsigned integer value in the "This Reg#/Value" column rather than a register number. These values will be displayed as integer for most operations, but will be displayed in hexadecimal for pack, fill, and unpack operations since these operate primarily on bit mask values.

The result of a test or compare will be zero if false, or one(s) if true. The true value will be the maximum unsigned 16-bit or 32-bit integer value if the result register is integer. If displayed as unsigned hexadecimal, it will be FFFF or FFFFFFFF. Displayed as signed 16-bit integer, "true" will be 32767. If the result register is floating point, then "true" will just be 1.0. The purpose of using FFFF for unsigned integer true is so that the result is useful as a bitmask.

Pack and fill are used for packing multiple local registers into a single register for purposes of emulating existing equipment when the ValuPoint is functioning as a Modbus server (slave). When pack and fill are used, "using" should be selected, and the second entry is a hexadecimal mask or fill value.

The pack mask is both a bit mask and position indicator. To calculate the contribution of a given calculate rule, the mask is right shifted until the least significant bit is nonzero, then this shifted mask is logically AND-ed with the local register content. The resulting masked value is then left shifted back to the original mask position. This final shifted result is then logically OR-ed into the result register (after first clearing the bits in the affected position of the result register).

Fill is simple - it simply logically OR's the bit mask into the result register.

Local Registers		Calculate	Copy	Report		
Showing 1 to 8 of 8						Update < Prev Next >
Rule #	Perform Operation	Using Register #	And/Thru Using	This Reg#/Value	Place Result in Register #	
1	pack	28	using	Fh	27	
2	pack	29	using	F0h	27	
3	pack	30	using	F00h	27	
4	fill	27	using	3000h	27	
5	pack	37	using	FFh	35	
6	pack	39	using	FF0000h	35	
7	fill	35	using	80000000h	35	
8	none	0	and	0	0	
# Rules Enabled: 8						Insert Delete

The example below shows the content of result registers 27 and 35 using the above calculate rules and the local register values shown. The contents of registers 28, 29, and 30 are packed into register 27. The contents of registers 37 and 39 are packed into 35 (all 32-bit register pairs).

Local Registers		Calculate	Copy	Report		
Showing registers from 27						Update < Prev Next >
Local Register #	Register Name	Set	Register Data		Register Format	
00027	Single Register 1	<input type="checkbox"/>	3432		Unsigned 16-bit	
00028	Single Register 2	<input type="checkbox"/>	2		Unsigned 16-bit	
00029	Single Register 3	<input type="checkbox"/>	3		Unsigned 16-bit	
00030	Single Register 4	<input type="checkbox"/>	4		Unsigned 16-bit	
00031	Single Register 5	<input type="checkbox"/>	0		Unsigned 16-bit	
00032	Single Register 6	<input type="checkbox"/>	0		Unsigned 16-bit	
00033	Single Register 7	<input type="checkbox"/>	0		Unsigned 16-bit	
00034	Single Register 8	<input type="checkbox"/>	0		Unsigned 16-bit	
00035	Double Register 1	<input type="checkbox"/>	807F003F		Unsigned 32-bit	
00037	Double Register 2	<input type="checkbox"/>	63		Unsigned 32-bit	
00039	Double Register 3	<input type="checkbox"/>	127		Unsigned 32-bit	
00041	Double Register 4	<input type="checkbox"/>	0		Unsigned 32-bit	

This process can be reversed using the "unpack" operation. The following calculate rules exactly reverse the above packing operations (discarding fill in this case).

Local Registers		Calculate	Copy	Report		
Showing 1 to 6 of 6						Update < Prev Next >
Rule #	Perform Operation	Using Register #	And/Thru Using	This Reg#/Value	Place Result in Register #	
1	unpack	27	using	Fh	28	
2	unpack	27	using	F0h	29	
3	unpack	27	using	F00h	30	
4	unpack	35	using	FFh	37	
5	unpack	35	using	FF0000h	39	
6	none	0	and	0	0	
# Rules Enabled: 6						Insert Delete

Register 27 is unpacked into registers 28, 29 and 30. Register 35 is unpacked into registers 37 and 39 (all 32-bit register pairs).

Local Registers		Calculate	Copy	Report		
Showing registers from 27						Update < Prev Next >
Local Register #	Register Name	Set	Register Data		Register Format	
00027	Single Register 1	<input type="checkbox"/>	3432		Unsigned 16-bit	
00028	Single Register 2	<input type="checkbox"/>	2		Unsigned 16-bit	
00029	Single Register 3	<input type="checkbox"/>	3		Unsigned 16-bit	
00030	Single Register 4	<input type="checkbox"/>	4		Unsigned 16-bit	
00031	Single Register 5	<input type="checkbox"/>	0		Unsigned 16-bit	
00032	Single Register 6	<input type="checkbox"/>	0		Unsigned 16-bit	
00033	Single Register 7	<input type="checkbox"/>	0		Unsigned 16-bit	
00034	Single Register 8	<input type="checkbox"/>	0		Unsigned 16-bit	
00035	Double Register 1	<input type="checkbox"/>	807F003F		Unsigned 32-bit	
00037	Double Register 2	<input type="checkbox"/>	63		Unsigned 32-bit	
00039	Double Register 3	<input type="checkbox"/>	127		Unsigned 32-bit	
00041	Double Register 4	<input type="checkbox"/>	0		Unsigned 32-bit	

The next two screen shots illustrate compare, set, and skip operations. Rule 5 says that rule 6 will not be executed if register 32 contains a zero. If register 32 is not equal to zero, then rule 6 will be executed.

Local Registers		Calculate	Copy	Report	
Showing 1 to 8 of 8				Update	< Prev Next >
Rule #	Perform Operation	Using Register #	And/Thru Using	This Reg#/Value	Place Result in Register #
1	comp = N	27	using	10	28
2	comp < N	27	using	10	29
3	comp > N	27	using	10	30
4	set	30	using	202	31
5	skip = N	32	using	0	32
6	set	33	using	0	33
7	set	34	using	88	34
8	none	0	and	0	0
# Rules Enabled: 8				Insert	Delete

Register values for examples using the above operations are illustrated below.

Local Registers		Calculate	Copy	Report	
Showing registers from 27				Update	< Prev Next >
Local Register #	Register Name	Set	Register Data	Register Format	
00027	Single Register 1	<input type="checkbox"/>	10	Unsigned 16-bit	
00028	Single Register 2	<input type="checkbox"/>	65535	Unsigned 16-bit	
00029	Single Register 3	<input type="checkbox"/>	0	Unsigned 16-bit	
00030	Single Register 4	<input type="checkbox"/>	0	Unsigned 16-bit	
00031	Single Register 5	<input type="checkbox"/>	202	Unsigned 16-bit	
00032	Single Register 6	<input type="checkbox"/>	0	Unsigned 16-bit	
00033	Single Register 7	<input type="checkbox"/>	0	Unsigned 16-bit	
00034	Single Register 8	<input type="checkbox"/>	88	Unsigned 16-bit	
00035	Double Register 1	<input type="checkbox"/>	0	Unsigned 32-bit	

4.5 Local Register Copy Rules

The copy rules provide a means of simply copying the content of one register to another.

ValuPoint IoT
MODEL VP6-1460
IOT EDGE SERVER

CONTROL SOLUTIONS MINNESOTA

Local Data Modbus IoT Cloud System

Data Events

Local Registers Calculate Copy Report

Showing 1 to 2 of 2

Rule #	Source Register #	Destination Register #
1	27	29
2	0	0

Rules Enabled: 2

The above rule would cause the following data copy to occur:

Local Registers Calculate Copy Report

Showing registers from 27

Local Register #	Register Name	Set	Register Data	Register Format
00027	Data Value 1	<input type="checkbox"/>	123.000000	Single Float
00029	Data Value 2	<input type="checkbox"/>	123.000000	Single Float

Here, however, is a much more interesting use of the copy rule:

Local Registers Calculate Copy Report

Showing 1 to 2 of 2

Rule #	Source Register #	Destination Register #
1	45	47
2	0	0

Rules Enabled: 2

The above rule would cause the following copy to happen. Note that "copy" also means data reformatting. Therefore, if you need to convert a number to a character string (or vice versa), simply copy it from one register to the other. In this example, our copy rule is converting floating point to an ASCII string ready to be sent to a display device.

Local Registers Calculate Copy Report

Showing registers from 45

Local Register #	Register Name	Set	Register Data	Register Format
00045	Data Value 10	<input type="checkbox"/>	45.980000	Single Float
00047	Char String 1	<input type="checkbox"/>	45.980000	Char String[40]

String conversion is not the only conversion you can do. If you need to convert floating point to integer, or vice versa, the copy rule will also do that. Note, however, that if you need to read an integer from a remote Modbus slave but want the result stored locally as floating point, you can do that conversion as part of the read map and do not need a separate copy rule to accomplish that conversion.

One more note about string conversion: If you do not like the appearance of numeric strings provided by the automatic conversion, you have full control over formatting if you use Script Basic to do the string conversion. Script Basic would also allow you to add text to the result, so you may end up with something like "Tank Level: 46 ft." instead of the "45.980000" in this example.

4.6 Device Status Reporting

The ValuPoint read maps include the ability to set a default value upon 'n' read fails, meaning that if the ValuPoint gets an error 'n' times attempting to read that point, it will automatically set the corresponding local register to the given default value to indicate the problem. This indication applies on a point by point basis, but of course any one point can be used as an indication that the entire device may be offline.

The ValuPoint also includes the ability to report device errors to an assigned status register rather than rely on default values. This reporting is configured on the Report page.

The screenshot shows the ValuPoint IoT Edge Server interface for Model VP6-1460. The 'Report' tab is active, displaying a table of device error reporting configurations. The table has columns for Report Status of, Device or Unit #, To This Register, With This Delay (Sec.), and Delete. Below the table is an 'Add' button with input fields for device type, unit number, register, and delay.

Report Status of	Device or Unit #	To This Register	With This Delay (Sec.)	Delete
Modbus RTU Master	1	27	20	<input type="checkbox"/>
Modbus RTU Master	2	28	20	<input type="checkbox"/>
Modbus TCP Client	1	29	20	<input type="checkbox"/>

Below the table, there is an 'Add' button with the following input fields: Modbus TCP Client (dropdown), 1 (text), 29 (text), 20 (text), and Add (button).

This optional list allows reporting device errors as register values to make it easier to monitor communication failures. The length of the list is variable. To add to the list, select the type of device to report on, select the device instance or unit number to report on, and select a register in which to put the status indication. Enter a delay if desired, and then click Add.

The delay is optional. If zero, there is no delay. If some number of seconds is entered,

then the error condition will not be reported until this time period expires. If the error clears before the time is up, then the error is never reported. This is useful for spurious errors that would result in nuisance indications.

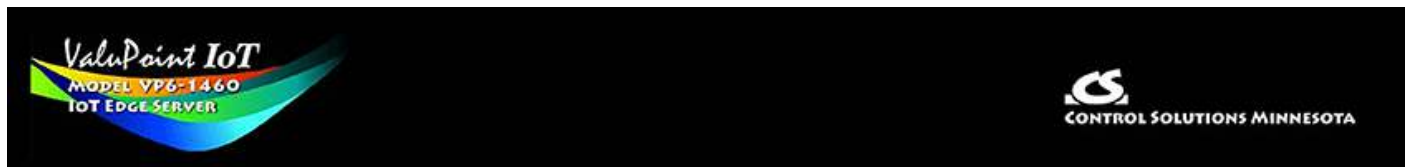
To remove a report from the list, check the box in the Delete column and then click Update. Click Prev or Next to scroll through the list.

Error codes placed into the reporting register will be as follows:

- 0 = No error
- 1 = Timeout, no response from remote device
- 2 = Error message received from remote device (e.g. Modbus exception)
- 3 = Line fault (e.g. CRC error, socket connection error, etc)

Once a Timeout error indication has been set (following delay if applicable), it will automatically return to zero upon the next successful communication with that device.

Once either the error message or line fault indication has been set, following delay if applicable, communication must continue free of this same error condition for at least the same delay period before the indication will be reset to zero. If an error message (e.g. Modbus exception) is reported for one data point, but multiple others are error free, then the one error would be hidden without this delay before reset. Ideally, this delay period should be at least as great as the poll period for the slowest point mapped.



5. Configuring ValuPoint as a Modbus RTU Master

The ValuPoint can be a Modbus RTU master or slave. As a master you can read Modbus data from, or write Modbus data to, other Modbus slaves. The ValuPoint will periodically poll the other Modbus devices according to register maps you set up. To read from a remote Modbus device, configure a Read Map. To write to a remote Modbus device, configure a Write Map.

Data read from a remote device is stored in a local register when received. Data written to a remote device is taken from a local register when sent. The local registers are the same collection of registers that are accessible to the MQTT engine for AWS IoT or Thingsboard interaction.

The following section details the process of setting up read and write maps via the web interface. Once familiar with map content, you have the option of importing bulk configurations as a CSV file. The import function is found on the File Manager page, and details about the content and format of the CSV file are found in Appendix B.

5.1 Modbus RTU Device Configuration

Modbus device configuration for RTU really consists of just port configuration. When brand new out of the box, the RTU port defaults to disabled. When disabled, Script Basic has access to the RS485 port for serial communication with non-Modbus devices.

The screenshot shows the configuration interface for a ValuPoint IoT Edge Server (Model VP6-1460) from Control Solutions Minnesota. The interface is divided into several sections:

- Navigation Tabs:** Local Data, Modbus, IoT Cloud, System, RTU Setup, RTU Data, TCP Setup, TCP Data, Local Device, RTU Read Map, RTU Write Map.
- Local Device Section:** Contains an "Update" button.
- Configuration Fields:**
 - Baud Rate: RTU disabled (dropdown)
 - Parity: None, 1 Stop Bit (dropdown)
 - I am the Master: (selected)
 - I am a Slave: (unselected)
 - Parameters for RTU Master:
 - Default Poll Rate: 0.000 Seconds (input field)
 - Timeout: 0.000 Seconds (input field)
 - Parameters for RTU Slave:
 - My Address or Unit #: 0 (input field)
 - Use FC 5/6 instead of 15/16 for unit numbers (slave addresses) starting at 0 (input field)

Select baud rate and parity from the drop down list. Click either Master or Slave buttons to select type of operation. Enter timing parameters or address as applicable. Click update to register your changes.

The default poll rate entered here will be used for all Modbus RTU Read and Write maps unless a different number is entered in the expanded view of the map.

IMPORTANT: Set timeout to something long enough for the device. If too short, the ValuPoint will not wait long enough for a response from the Modbus slave device, and the result will be a lot of "no response" errors from the device even though the device is perfectly functional.

If your slave/server device only supports function codes 5 and 6 for writing coils and holding registers, check the Use FC 5/6 box. The default function codes are 15 and 16, which are most widely used. If you check the box, you should also enter a "starting at" unit # or slave address. This allows supporting both types of devices at the same time provided you assign slave addresses in two non-overlapping groups. (These settings do not apply if the ValuPoint is the slave.)

5.2 Modbus RTU Master Read Maps

Getting the ValuPoint to read registers from another Modbus device requires setting up a "Read Map" as shown here.

Map #	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Local Register #	Name
1	None	None	0	0	0	

Map number simply tells you where you're at on the list of register maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Showing" box, then click Update.

Maps entered on this page only read data from remote devices. Go to the RTU Write Map to write data to those devices. The full parameter set is different for read versus write.

Local Device		RTU Read Map		RTU Write Map			
Showing 1 to 1 of 1							
Update < Prev Next >							
Map #	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Local Register #	Name	
1	None	None	0	0	0		
<div style="display: flex; align-items: flex-start;"> <div style="border: 1px solid black; padding: 2px; margin-right: 5px;">Quick</div> <div style="border: 1px solid black; padding: 2px;"> None Coil (output) Discrete Input Input Register Holding Register </div> </div> <p>This page only applies if the local device is configured as a Master.</p>							

To create a Read Map, start by selecting the Modbus register type to read from the drop-down list.

Local Device		RTU Read Map		RTU Write Map			
Showing 1 to 1 of 1							
Update < Prev Next >							
Map #	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Local Register #	Name	
1	Holding Register	None	0	0	0		
<div style="display: flex; align-items: flex-start;"> <div style="border: 1px solid black; padding: 2px; margin-right: 5px;">Quick Help</div> <div style="border: 1px solid black; padding: 2px;"> None INT-16 UINT-16 INT-32 UINT-32 INT-64 UINT-64 FLOAT DOUBLE CHAR MOD10-2 MOD10-3 MOD10-4 </div> </div> <p>This page only applies if the local device is configured as a Master.</p> <p>Map number simply tells you where the register is located in the list of register maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Map #" field, then click Update.</p> <p>Maps entered on this page only read from the remote devices. Go to the RTU Write Map to write data to those devices. The full parameter set is different for read versus write.</p> <p>An abbreviated version of a list of Modbus register types is shown in this page. Any of the parameters shown may be changed here and registered by clicking the Update button. To view and/or modify the complete set of parameters, click on the map number in the left most column.</p>							

Select the data format expected in the remote Modbus register. The abbreviation INT under Format means signed integer, while UINT represents unsigned integer. The INT and UINT are followed by the number of bits to be read (which translates into 1, 2, or 4 consecutive holding registers). The FLOAT format refers to 32-bit IEEE 754 format while DOUBLE refers to 64-bit IEEE 754 floating point. The MOD10 format is unique to Schneider Electric power meters, and is supported in 2, 3, and 4-register formats. (Note: Use INT-16 or UINT-16 for coils or discrete inputs - in this case format only affects local register data conversion.)

Local Device		RTU Read Map		RTU Write Map			
Showing 1 to 2 of 2							
Update < Prev Next >							
Map #	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Local Register #	Name	
1	Holding Register	UINT-16	11	2	27	Data Value 1	
2	None	None	0	0	0		

Enter the register number to read from the remote RTU slave and slave address of that RTU slave. Do not use Modicon numbers here. In other words, if your slave

device's documentation says read register 40001, that is short hand (Modicon notation) for saying read holding register 1. Refer to the Modbus Reference Information section of this user guide for more discussion about register numbers like 40001. If you enter 40001 here to read the first holding register, you will get an exception error since the actual register number is not 40001.

The Local Register is where data read from the remote Modbus RTU slave will be stored locally in the ValuPoint. If the local register data format does not match what you are reading from the Modbus slave, the data will be converted automatically when it is read.

Local Device		RTU Read Map		RTU Write Map					
Showing 1 to 2 of 2							Update	< Prev	Next >
Map #	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Local Register #	Name			
1	Holding Register	UINT-16	11	2	27	Data Value 1			
	None	None	0	0	0				

Click on the Map number in the first column to access the expanded view of the Read Map.

Local Device		RTU Read Map		RTU Write Map					
Map # 1							Update	< Prev	Next >
Read Holding Register as Unsigned 16-bit Size: 0									
From register # 11 at Unit # 2 With low register first if checked: <input type="checkbox"/>									
Apply bit mask if applicable: 0000 then apply scale: 0.000000 and offset: 0.000000									
Save in local register # 27 named Data Value 1 Repeat this process every 5.0 seconds.									
Apply this default value: 0.000000 after 0 read failure(s).									
<input type="checkbox"/> Enable this map only when index register 0 is set to a value of 0									
# RTU Read Maps Enabled: 2							Insert	Delete	

For each remote register to be read, select the register type, format, number, and remote unit (slave address). The optional bit mask and scaling are discussed with examples below.

Modbus protocol treats all input registers or holding registers as strictly 16-bit registers. To accommodate 32-bit or longer data, Modbus devices use multiple consecutive "registers" to hold the data. There is no standardization of whether the least significant part of the data comes first or last. Therefore, ValuPoint lets you set that according to whatever the slave device requires. If the least significant data is found in the first (or lower numbered) register in your slave device, then check the box after "With low register first".

The poll rate ("Repeat this process...") determines how often the remote register will be read. If zero is entered here, the rate will become the default poll rate given on the

Devices page for the Modbus RTU port.

The default value will be stored into the local register after the given number of read failures if the fail count is non-zero. Setting the count to zero will disable the default, and the object will retain the most recent value obtained. If the default value does take effect, the actual data value read will be retained when communications are restored.

You have the option of making this Read Map conditional. If an index register number is provided and the Enable box is checked, then this read map will only be executed when the index register (local register) contains the value given. This allows multiple read maps to supply data to the same local register based on the value of the index register. It also allows reading to simply be suspended if a single read map supplies data to the local register. In a more sophisticated scenario, you could potentially suspend reading of the slave if you know the slave is powered down.

Map number simply tells you where you're at on the list of register maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Map #" box, then click Update.

Delete will remove the map number shown in the "Map #" box. Insert will insert a new map before the map number shown, and is used for placing maps between existing maps. It is not necessary to use Insert to add maps to the bottom of the list or to define any map presently having zero for a source object or "none" for remote type.

Selecting "none" for remote type effectively deletes the map even though it will still appear in the list until deleted. Unused maps at the end of the list will always show none as the type. If you wish to prevent these from being displayed, reduce the number of maps enabled.

The number of maps enabled simply limits the scope of map review so that you do not have to review a lot of unused maps. If the displayed maps are used up and you need more, increase the enabled number.

The screenshot shows the 'RTU Read Map' configuration window. At the top, there are tabs for 'Local Device', 'RTU Read Map', and 'RTU Write Map'. The 'RTU Read Map' tab is active. Below the tabs, there is a 'Map #' input field containing the number '1'. To the right of this field are 'Update', '< Prev', and 'Next >' buttons. The main configuration area is a dark teal panel with the following fields and controls:

- 'Read' dropdown menu set to 'Holding Register', 'as' dropdown set to 'Unsigned 16-bit', and 'Size' input field set to '0'.
- 'From register #' input field set to '11', 'at Unit #' input field set to '2', and a checkbox for 'With low register first if checked:' which is unchecked.
- 'Apply bit mask if applicable:' input field set to '0000', 'then apply scale:' input field set to '1.800000', and 'and offset:' input field set to '32.000000'.
- 'Save in local register #' input field set to '27', 'named' input field set to 'Data Value 1', and 'Repeat this process every' input field set to '5.0' seconds.
- 'Apply this default value:' input field set to '0.000000', 'after' input field set to '0', and 'read failure(s)'.
- A checkbox for 'Enable this map only when index register' which is unchecked, followed by an input field set to '0' and 'is set to a value of' followed by an input field set to '0'.

At the bottom of the configuration area, there is a '# RTU Read Maps Enabled:' input field set to '2'. To the right of this field are 'Insert' and 'Delete' buttons.

You have the option of providing a scale and offset. A scale of zero will cause scale and offset to be ignored. If provided, the Modbus data will be treated as raw data. When

the Modbus data is received, it will be multiplied by scale, then added to offset, and then stored in the local register. If the Modbus slave was providing degrees Celsius, and the scale factors illustrated above were used, then a Modbus value of 25 would result in the local register receiving a value of 77 (degrees Fahrenheit).

Local Register #	Register Name	Set	Register Data	Register Format
00027	Data Value 1	<input type="checkbox"/>	77	Unsigned 16-bit
00028	Data Value 2	<input type="checkbox"/>	0	Unsigned 16-bit
00029	Data Value 3	<input type="checkbox"/>	0	Unsigned 16-bit

It is common for Modbus devices to pack a number of status bits into a single holding register. In order to do meaningful things based on a single bit, it is sometimes necessary to split that register into multiple local registers. ValuPoint supports this requirement by providing an optional bit mask.

If a bit mask is entered (in hexadecimal), and the remote register type is signed or unsigned integer (16-bit or 32-bit data), the mask will be bit-wise logical AND-ed with the Modbus data, and the retained bits will be right justified in the result stored locally.

Local Device	RTU Read Map	RTU Write Map
Map # 1		
Read Holding Register as Unsigned 16-bit Size: 0		
From register # 11 at Unit # 2 With low register first if checked: <input type="checkbox"/>		
Apply bit mask if applicable: 0001 then apply scale: 0.000000 and offset: 0.000000		
Save in local register # 27 named Data Value 1 Repeat this process every 5.0 seconds.		
Apply this default value: 0.000000 after 0 read failure(s).		
<input type="checkbox"/> Enable this map only when index register 0 is set to a value of 0		
# RTU Read Maps Enabled: 5		

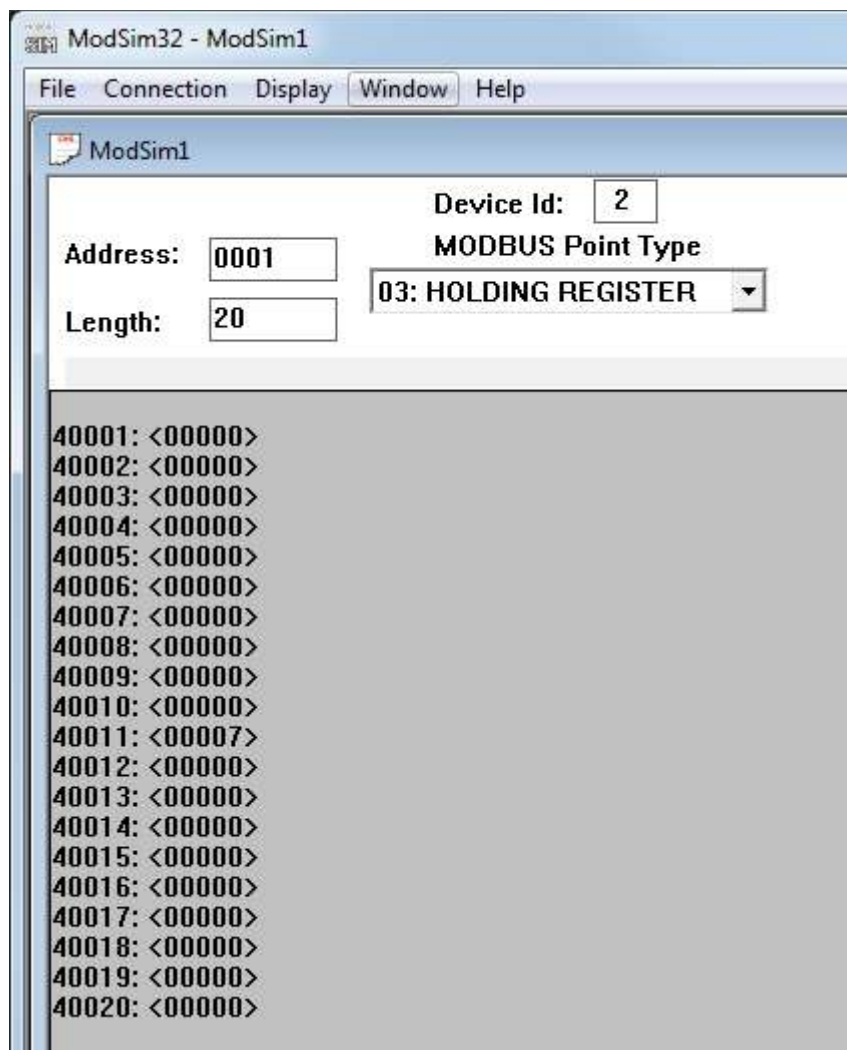
Reading bit 0 is illustrated above while reading bit 3 is illustrated below (bit 3 mask in binary would be 1000 which is hexadecimal 8 as entered in the configuration page). Refer to the Modbus Reference section in this user guide for a list of all possible mask values.

Local Device	RTU Read Map	RTU Write Map			
Map #	4			Update	< Prev Next >
Read	Holding Register	as	Unsigned 16-bit	Size:	0
From register #	11	at Unit #	1	With low register first if checked:	<input type="checkbox"/>
Apply bit mask if applicable:	0008	then apply scale:	0.000000	and offset:	0.000000
Save in local register #	30	named	Data Value 4	Repeat this process every	5.0 seconds.
Apply this default value:	0.000000	after	0	read failure(s).	
<input type="checkbox"/>	Enable this map only when index register	0	is set to a value of	0	
# RTU Read Maps Enabled:	5			Insert	Delete

If the read maps referencing the same remote register are created in sequential contiguous order, the ValuPoint will optimize the RTU activity by reading the remote register once and then sharing the data with all of the read maps in the group. In the example illustrated here, four consecutive read maps reference the same remote register, each selecting a different bit. The first map is selecting bit 0, the second selecting bit 1, and so on.

Local Device	RTU Read Map	RTU Write Map				
			Showing	1	to 5 of 5	Update < Prev Next >
Map #	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Local Register #	Name
1	Holding Register	UINT-16	11	2	27	Data Value 1
2	Holding Register	UINT-16	11	2	28	Data Value 2
3	Holding Register	UINT-16	11	1	29	Data Value 3
4	Holding Register	UINT-16	11	1	30	Data Value 4
5	None	None	0	0	0	

To try out these read maps, we have set up ModSim with the remote register 11 containing a value of 7.



The holding register value of 7 translates into the following local register values when the bit mask option is used as illustrated.

Local Registers		Calculate	Copy	Report	
Showing registers from 27				Update	< Prev Next >
Local Register #	Register Name	Set	Register Data	Register Format	
00027	Data Value 1	<input type="checkbox"/>	1	Unsigned 16-bit	
00028	Data Value 2	<input type="checkbox"/>	1	Unsigned 16-bit	
00029	Data Value 3	<input type="checkbox"/>	1	Unsigned 16-bit	
00030	Data Value 4	<input type="checkbox"/>	0	Unsigned 16-bit	

5.3 Modbus RTU Master Write Maps

Getting the ValuPoint to write registers to another Modbus device requires setting up a "Write Map" as shown here. Much of the Write Map is configured the same as a Read Map.

Local Device		RTU Read Map		RTU Write Map			
Showing 1 to 2 of 2							
Update < Prev Next >							
Map #	Local Register #	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Name	
1	28	Holding Register	UINT-16	12	1	Data Value 2	
2	0	None	None	0	0		

The data direction is reversed but the same selections are still made. Select the local register that will be the source of data to write to the remote Modbus RTU slave. Select the register type, data format, and register number to be written to in that slave. Enter the slave address as Remote Unit. Click on the map number in the first column to access additional optional configuration parameters.

Local Device		RTU Read Map		RTU Write Map			
Map # 1							
Update < Prev Next >							
Read local register # 28 named Data Value 2							
Write remote register <input checked="" type="checkbox"/> when local register changes by > 0.000000 or <input type="checkbox"/> when 0.0 seconds have elapsed with no change.							
Otherwise write remote register unconditionally, applying local register data as follows:							
Apply scale: 0.000000 and offset: 0.000000 Then if applicable, apply bit mask: 0000 and bit fill: 0000							
Write Holding Register as Unsigned 16-bit Size: 0 with blank padding if checked <input type="checkbox"/>							
To register # 12 at Unit # 1 With low register first if checked: <input type="checkbox"/>							
Repeat this process <input type="radio"/> at least <input checked="" type="radio"/> no more than every 0.0 seconds.							
<input type="checkbox"/> Enable this map only when index register 0 is set to a value of 0							
# Client Write Maps Enabled: 2							
Insert Delete							

The local register data may be written to the remote slave periodically, or when it changes, or both. To send upon change (send on delta), check the first box and enter the amount by which the local register must change before being written to the remote device. To guarantee that the remote register will be written at least occasionally even if the data does not change, check the second box and enter some amount of time. This time period will be referred to as the "maximum quiet time".

Data from the local register may be manipulated before being written to the remote register. The local data is first multiplied by the scale factor. The offset is then added to it. If a bit mask is entered, and the remote register type is signed or unsigned integer (16-bit or 32-bit data), the mask will be bit-wise logical AND-ed with the data. The mask is right justified, then AND-ed with the data. The result is then left shifted back to the original position of the mask. In other words, the least significant bits of the original data will be stuffed at the position marked by the mask.

After the scaling and masking, the bit fill will be logically OR-ed into the result, but only if the mask was nonzero and was used. Both mask and fill are entered in hexadecimal. The effect of "fill" is that certain bits will always be set to 1 in the data written to the remote Modbus device.

Multiple local registers may be packed into a single remote register. To accomplish this, define two or more maps in sequence with the same remote destination. If the destination is the same, data types are 16 or 32-bit integer (signed or unsigned), bit masks are nonzero, and the maps are sequential, the results of all qualifying maps will be OR-ed together before being sent to the remote destination.

For the remote register to be written, select the register type, format, number, and remote unit (slave address). Data formats are the same as described above for Read Maps. Size is only specified for character strings. Use INT-16 or UINT-16 data format for coils - in this case format only affects local register data conversion.

Modbus protocol treats all holding registers as strictly 16-bit registers. To accommodate 32-bit or longer data, Modbus devices use multiple consecutive "registers" to hold the data. There is no standardization of whether the least significant part of the data comes first or last. Therefore, ValuPoint lets you set that according to whatever the slave device requires. If the least significant data is found in the first (or lower numbered) register in your slave device, then check the box after "With low register first".

The repeat time may determine how often the remote register will be written. If send on delta and maximum quiet time are not checked above, clicking the "at least" button will establish a periodic update time. If send on delta is used and you wish to limit the network traffic when changes are too frequent, click the "no more than" button and enter the minimum time that should elapse before allowing another write to the remote device. It is valid to select "no more than every 0.0 seconds" if you want all changes to be sent, but no periodic writes.

You have the option of making this Write Map conditional. If an index register number is provided and the Enable box is checked, then this write map will only be executed when the index register (local register) contains the value given. This allows multiple write maps to supply data to the same remote register based on the value of the local index register. It also allows writing to simply be suspended if a single write map supplies data to the remote register. In a more sophisticated scenario, you could potentially suspend writing of the slave if you know the slave is powered down.

Delete will remove the map number shown in the "Map #" box. Insert will insert a new map before the map number shown, and is used for placing maps between existing maps. It is not necessary to use Insert to add maps to the bottom of the list or to define any map presently having zero for a source register or "none" for remote type.

Selecting "none" for remote type effectively deletes the map even though it will still appear in the list until deleted. Unused maps at the end of the list will always show none as the type. If you wish to prevent these from being displayed, reduce the number of maps enabled.

5.4 Modbus RTU Master Data Displayed by Slave

The RTU Registers page shows a list of local registers mapped to RTU slave devices. The page will show only one device at a time, and may have no entries as illustrated below.

The screenshot shows the 'RTU Registers' page for RTU Unit # 1. The interface includes navigation tabs for RTU Setup, RTU Data, TCP Setup, and TCP Data. Under RTU Data, there are sub-tabs for Error Counts, Errors: Read Maps, and Errors: Write Maps. The 'RTU Registers' sub-tab is active. The page shows 'Showing 1 to 0 of 0' registers. A table with 8 columns is displayed: Dir, Reg. Type, Remote Reg. #, Register Name, Local Reg. #, Update, Register Data, and Time since Last update. The table contains one row with all fields set to '---'. At the bottom, there is a control for 'RTU Unit # 1' and buttons for '< Prev Unit' and 'Next Unit >'.

Dir.	Reg. Type	Remote Reg. #	Register Name	Local Reg. #	Update	Register Data	Time since Last update
---	Undefined	---		---	<input type="checkbox"/>	---	---

Click the Next Unit button to go to the next RTU slave, or simply enter a number in the RTU Unit # window and click Update. Registers for that slave will now be displayed. In addition to a summary of the map (both read and write maps are shown), the time since last update is displayed. This time should generally be less than the poll time. If the last update time is large, it means there is an error preventing the update.

The screenshot shows the 'RTU Registers' page for RTU Unit # 1. The interface is similar to the previous screenshot, but the 'Showing' indicator now says 'Showing 1 to 3 of 3'. The table contains three rows of data:

Dir.	Reg. Type	Remote Reg. #	Register Name	Local Reg. #	Update	Register Data	Time since Last update
From	Holding Reg	00011	Data Value 3	00029	<input type="checkbox"/>	1	540374.000
From	Holding Reg	00011	Data Value 4	00030	<input type="checkbox"/>	0	540374.000
To	Holding Reg	00012	Data Value 2	00028	<input type="checkbox"/>	1	540374.000

5.5 Modbus RTU Errors

The Error Counts page shows a tabulation, by RTU slave, of all errors observed. In the example below, we can see that the RTU device at slave address 1 is running flawlessly while slave #2 has had some issues.

ValuPoint IoT
MODEL VP6-1460
IOT EDGE SERVER

CONTROL SOLUTIONS MINNESOTA

Local Data | Modbus | IoT Cloud | System

RTU Setup | RTU Data | TCP Setup | TCP Data

RTU Registers | **Error Counts** | Errors: Read Maps | Errors: Write Maps

Showing devices from

Unit #	Reset	Total Messages	No Responses	CRC Errors	Exceptions
1	<input type="checkbox"/>	102007	0	0	0
2	<input type="checkbox"/>	683	457	0	226
3	<input type="checkbox"/>	0	0	0	0
4	<input type="checkbox"/>	0	0	0	0

If the counts show some problems, we can look for more detail on the Errors: Read Maps (or Errors: Write Maps) pages. These pages will tell us exactly which Read Map (or Write Map) the problem is occurring on, and what the error is, as illustrated below.

RTU Registers | Error Counts | **Errors: Read Maps** | Errors: Write Maps

Map #	Register Name	Error Description	Exception Code
1	Data Value 1	Exception code returned by device	Illegal data address

When the problem is resolved, it will be removed from this list.

RTU Registers | Error Counts | **Errors: Read Maps** | Errors: Write Maps

Map #	Register Name	Error Description	Exception Code
--	---	---	---

Once you have resolved problems, you can reset the error counts by checking the box in the Reset column and then clicking Update.

RTU Registers | **Error Counts** | Errors: Read Maps | Errors: Write Maps

Showing devices from

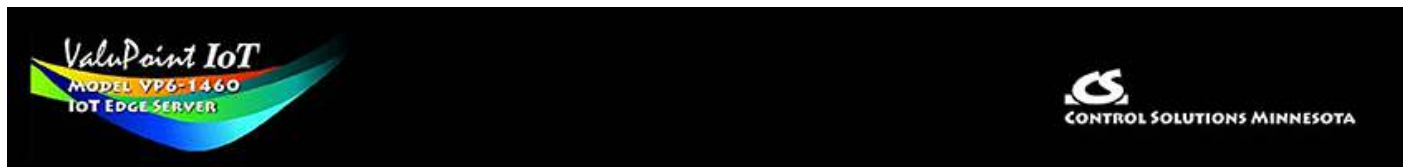
Unit #	Reset	Total Messages	No Responses	CRC Errors	Exceptions
1	<input checked="" type="checkbox"/>	102007	0	0	0
2	<input checked="" type="checkbox"/>	1805	457	0	1328
3	<input type="checkbox"/>	0	0	0	0

The counts will reset to zero, but in most cases, at least "Total Messages" will start

incrementing again.

RTU Registers		Error Counts		Errors: Read Maps		Errors: Write Maps	
Unit #	Reset	Total Messages	No Responses	CRC Errors	Exceptions		
1	<input type="checkbox"/>	0	0	0	0		
2	<input type="checkbox"/>	3	0	0	0		
3	<input type="checkbox"/>	0	0	0	0		

Showing devices from



6. Configuring ValuPoint as a Modbus TCP Client

The ValuPoint can be a Modbus TCP client and server. The terms client and server are more often used with Ethernet network devices, but for Modbus purposes, they still mean master and slave respectively. You must choose one or the other between master and slave for Modbus RTU, but Modbus TCP can be both simultaneously thanks to Ethernet.

As a master (client) you can read Modbus data from, or write Modbus data to, other Modbus TCP devices. The ValuPoint will periodically poll the other Modbus devices according to register maps you set up. To read from a remote Modbus device, configure a Read Map. To write to a remote Modbus device, configure a Write Map.

Data read from a remote device is stored in a local register when received. Data written to a remote device is taken from a local register when sent. The local registers are the same collection of registers that are accessible to other Modbus TCP clients as a collection of holding registers. These local registers are also accessible to the MQTT engine for AWS IoT or Thingsboard interaction.

The following section details the process of setting up read and write maps via the web interface. Once familiar with map content, you have the option of importing bulk configurations as a CSV file. The import function is found on the File Manager page, and details about the content and format of the CSV file are found in Appendix B.

6.1 Modbus TCP Device Configuration

The Modbus TCP client is the Ethernet version of Modbus master. Modbus RTU requires a slave address. Modbus TCP also requires, in effect, a slave address. In the case of TCP, that slave address is an IP address. Since entering the IP address requires more effort than one simple slave number, and because internally a network socket is required per IP address, the TCP devices are set up in their own table.

For each Modbus TCP device that you wish to read and write, enter its IP address on the TCP Setup Devices page. The port is normally going to be 502 (the standard Modbus TCP port), but if it is different, enter that number here.

ValuPoint IoT
MODEL VP6-1460
IOT EDGE SERVER

CONTROL SOLUTIONS MINNESOTA

Local Data Modbus IoT Cloud System

RTU Setup RTU Data TCP Setup TCP Data

Devices Client Read Map Client Write Map

Device # 1 Update < Prev Next >

Local Name Device 1

Use Static IPv4 Static IPv6 Domain Lookup

IP Address 192.168.1.134 Port: 502

Domain Name

Unit (optional) 1 Use FC 5/6 instead of 15/16

Default Poll Period 5.0 Seconds Connection Status 0 Clear

A unit number is always included in each Modbus TCP packet. This is the equivalent of the RTU slave address. Some TCP devices pay no attention to unit and simply echo back whatever you had sent. However, if you are accessing RTU devices on the other side of a TCP to RTU router (gateway), then the unit does become the RTU slave address on the RTU side of the gateway and multiple RTU devices are accessed at the same TCP IP address.

If "Unit" on the devices page is left at zero, then unit #1 is used by default. Otherwise the number entered here becomes the default. However, each individual read and write map can have different unit numbers, as would be required for accessing multiple RTU devices via a gateway at one IP address. When a non-zero unit number is placed in the read or write map, it will override the default on the Devices page.

If coil and holding register writes must be done with write single function codes 5 and 6 rather than write multiple function codes 15 and 16, then check the "Use FC 5/6" box for this device. The ValuPoint will default to write multiple.

The default poll rate entered here will be used for all Modbus TCP Read and Write maps unless a different number is entered in the expanded view of the map.

The static IP address for the Modbus TCP device can be either IPv4 or IPv6. Along with selecting the desired IP version, be sure to enter the applicable IP address format. In addition, if IPv6 is used, be sure IPv6 is enabled on the Network configuration page.

The screenshot shows the configuration page for Device # 2. The interface has a dark green header with tabs for "Devices", "Client Read Map", and "Client Write Map". The "Devices" tab is active. Below the header, there are navigation buttons: "Update", "< Prev", and "Next >". The main configuration area is a dark teal background with white text and input fields. The "Local Name" is "Device 2". Under "Use", the "Static IPv6" radio button is selected. The "IP Address" is "fe80::c28:cf9b:b7f8:639e". The "Port" is "502". The "Domain Name" field is empty. The "Unit (optional)" is "1". There is a checkbox for "Use FC 5/6 instead of 15/16" which is unchecked. The "Default Poll Period" is "5.0" seconds. The "Connection Status" is "0" and there is a "Clear" button next to it.

You have the option of referring to a Modbus TCP device by domain name. If you use a domain name, be sure that domain can be found at the DNS servers provided on the Network setup page. If found, the IP address provided by DNS will be displayed here (but not saved in the XML configuration file).

The screenshot shows the configuration page for Device # 3. The interface is similar to the previous one. The "Local Name" is "Device 3". Under "Use", the "Domain Lookup" radio button is selected. The "IP Address" is "50.97.153.22". The "Port" is "502". The "Domain Name" is "myModbusDevice.net". The "Unit (optional)" is "1". There is a checkbox for "Use FC 5/6 instead of 15/16" which is unchecked. The "Default Poll Period" is "5.0" seconds. The "Connection Status" is "0" and there is a "Clear" button next to it.

6.2 Modbus TCP Client Read Maps

Getting the ValuPoint to read registers from another Modbus device requires setting up a "Read Map" as shown here.

ValuPoint IoT
MODEL VP6-1460
IOT EDGE SERVER

CONTROL SOLUTIONS MINNESOTA

Local Data Modbus IoT Cloud System

RTU Setup RTU Data TCP Setup TCP Data

Devices Client Read Map Client Write Map

Showing 1 to 1 of 1 Update < Prev Next >

Map #	Remote Type	Remote Register Format	Remote Register #	Remote Device	Local Register #	Name
1	None	None	0	None	0	

Map number simply tells you where you're at on the list of register maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Showing" box, then click Update.

Maps entered on this page only read data from remote devices. Go to the TCP Write Map to write data to those devices. The full parameter set is different for read versus write.

RTU Setup RTU Data TCP Setup TCP Data

Devices Client Read Map Client Write Map

Showing 1 to 1 of 1 Update < Prev Next >

Map #	Remote Type	Remote Register Format	Remote Register #	Remote Device	Local Register #	Name
1	None	None	0	None	0	

Quick

Read remote registers into local registers. This page creates a map entry that reads data from one or more remote Modbus/TCP servers for processing here. Click on map number to see more detail and insert/delete maps.

To create a Read Map, start by selecting the Modbus register type to read from the drop-down list.

Devices		Client Read Map		Client Write Map		
Showing 1 to 1 of 1						
Update < Prev Next >						
Map #	Remote Type	Remote Register Format	Remote Register #	Remote Device	Local Register #	Name
1	Holding Register	None	0	None	0	

Quick Help

Read remote registers into local registers. Click on map number to see more details.

Map number simply tells you where the data is stored. To create a map entry that reads data from one or more remote Modbus/TCP servers for processing here, click on the map number, then click Update.

Maps entered on this page only read data from remote devices. Go to the Client Write Map to write data to those devices. The full parameter set is different for read versus write.

An abbreviated version of a list of remote devices is shown on this page. Any of the parameters shown may be changed here and registered by clicking the Update button. To view and/or modify the complete set of parameters, click on the map number in the left most column.

Select the data format expected in the remote Modbus register. The abbreviation INT under Format means signed integer, while UINT represents unsigned integer. The INT and UINT are followed by the number of bits to be read (which translates into 1, 2, or 4 consecutive holding registers). The FLOAT format refers to 32-bit IEEE 754 format while DOUBLE refers to 64-bit IEEE 754 floating point. The MOD10 format is unique to Schneider Electric power meters, and is supported in 2, 3, and 4-register formats. (Note: Use INT-16 or UINT-16 for coils or discrete inputs - in this case format only affects local register data conversion.)

Devices		Client Read Map		Client Write Map		
Showing 1 to 2 of 2						
Update < Prev Next >						
Map #	Remote Type	Remote Register Format	Remote Register #	Remote Device	Local Register #	Name
1	Holding Register	UINT-16	1	ModSim	27	Data Value 1
2	None	None	0	None	0	

Enter the register number to read from the remote TCP server. Do not use Modicon numbers here. In other words, if your device's documentation says read register 40001, that is short hand (Modicon notation) for saying read holding register 1. Refer to the Modbus Reference Information section of this user guide for more discussion about register numbers like 40001. If you enter 40001 here to read the first holding register, you will get an exception error since the actual register number is not 40001.

Select a TCP device from the list that this register should be read from. Only devices entered on the Devices page will appear in the list.

The Local Register is where data read from the remote Modbus TCP server will be stored locally in the ValuPoint. If the local register data format does not match what you are reading from the Modbus device, the data will be converted automatically when it is read.

Devices		Client Read Map		Client Write Map					
Showing 1 to 2 of 2							Update	< Prev	Next >
Map #	Remote Type	Remote Register Format	Remote Register #	Remote Device	Local Register #	Name			
1	Holding Register	UINT-16	1	ModSim	27	Data Value 1			
2	None	None	0	None	0				

Click on the Map number in the first column to access the expanded view of the Read Map.

Devices		Client Read Map		Client Write Map					
Map # 1							Update	< Prev	Next >
Read <input type="text" value="Holding Register"/> as <input type="text" value="Unsigned 16-bit"/> Size: <input type="text" value="0"/>									
From register # <input type="text" value="1"/> at <input type="text" value="ModSim"/> unit # <input type="text" value="1"/> With low register first if checked: <input type="checkbox"/>									
Apply bit mask if applicable: <input type="text" value="0000"/> then apply scale: <input type="text" value="1.800000"/> and offset: <input type="text" value="32.000000"/>									
Save in local register # <input type="text" value="27"/> named <input type="text" value="Data Value 1"/> Repeat this process every <input type="text" value="5.0"/> seconds.									
Apply this default value: <input type="text" value="0.000000"/> after <input type="text" value="0"/> read failure(s).									
<input type="checkbox"/> Enable this map only when index register <input type="text" value="0"/> is set to a value of <input type="text" value="0"/>									
# Client Read Maps Enabled: <input type="text" value="2"/>							Insert	Delete	

For each remote register to be read, select the register type, format, number. Select a TCP server device from the list to read from. Only devices entered on the Devices page will appear here. If a unit number other than the default unit entered for this TCP server on the Devices page should be used, enter that unit number here.

The optional bit mask and scaling are discussed with examples below.

Modbus protocol treats all input registers or holding registers as strictly 16-bit registers. To accommodate 32-bit or longer data, Modbus devices use multiple consecutive "registers" to hold the data. There is no standardization of whether the least significant part of the data comes first or last. Therefore, ValuPoint lets you set that according to whatever the remote Modbus device requires. If the least significant data is found in the first (or lower numbered) register in your Modbus device, then check the box after "With low register first".

The poll rate ("Repeat this process...") determines how often the remote register will be read. If zero is entered here, the rate will become the default poll rate given on the Devices page for the Modbus TCP device selected.

The default value will be stored into the local register after the given number of read failures if the fail count is non-zero. Setting the count to zero will disable the default, and the object will retain the most recent value obtained. If the default value does take effect, the actual data value read will be retained when communications are restored.

You have the option of making this Read Map conditional. If an index register number is provided and the Enable box is checked, then this read map will only be executed when the index register (local register) contains the value given. This allows multiple read maps to supply data to the same local register based on the value of the index register. It also allows reading to simply be suspended if a single read map supplies data to the local register. In a more sophisticated scenario, you could potentially suspend reading of the remote Modbus device if you know the device is powered down.

Map number simply tells you where you're at on the list of register maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter the desired number in the "Map #" box, then click Update.

Delete will remove the map number shown in the "Map #" box. Insert will insert a new map before the map number shown, and is used for placing maps between existing maps. It is not necessary to use Insert to add maps to the bottom of the list or to define any map presently having zero for a source object or "none" for remote type.

Selecting "none" for remote type effectively deletes the map even though it will still appear in the list until deleted. Unused maps at the end of the list will always show none as the type. If you wish to prevent these from being displayed, reduce the number of maps enabled.

The number of maps enabled simply limits the scope of map review so that you do not have to review a lot of unused maps. If the displayed maps are used up and you need more, increase the enabled number.

Local Registers		Calculate	Copy	Report	
Showing registers from				27	Update < Prev Next >
Local Register #	Register Name	Set	Register Data	Register Format	
00027	Data Value 1	<input type="checkbox"/>	77	Unsigned 16-bit	
00028	Data Value 2	<input type="checkbox"/>	0	Unsigned 16-bit	
00029	Data Value 3	<input type="checkbox"/>	0	Unsigned 16-bit	

You have the option of providing a scale and offset. A scale of zero will cause scale and offset to be ignored. If provided, the Modbus data will be treated as raw data. When the Modbus data is received, it will be multiplied by scale, then added to offset, and then stored in the local register. If the Modbus device was providing degrees Celsius, and the scale factors illustrated above were used, then a Modbus value of 25 would result in the local register receiving a value of 77 (degrees Fahrenheit).

It is common for Modbus devices to pack a number of status bits into a single holding register. In order to do meaningful things based on a single bit, it is sometimes necessary to split that register into multiple local registers. ValuPoint supports this requirement by providing an optional bit mask.

If a bit mask is entered (in hexadecimal), and the remote register type is signed or unsigned integer (16-bit or 32-bit data), the mask will be bit-wise logical AND-ed with the Modbus data, and the retained bits will be right justified in the result stored locally.

Refer to the Modbus Reference section in this user guide for a list of all possible mask values.

If the read maps referencing the same remote register are created in sequential contiguous order, the ValuPoint will optimize the TCP activity by reading the remote register once and then sharing the data with all of the read maps in the group. The example illustrated for Modbus RTU in section 5.2 works exactly the same for TCP. In that example, four consecutive read maps reference the same remote register, each selecting a different bit. The first map is selecting bit 0, the second selecting bit 1, and so on.

6.3 Modbus TCP Client Write Maps

Getting the ValuPoint to write registers to another Modbus device requires setting up a "Write Map" as shown here. Much of the Write Map is configured the same as a Read Map.

Devices		Client Read Map		Client Write Map			
Showing 1 to 2 of 2				Update		< Prev Next >	
Map #	Local Register #	Remote Type	Remote Register Format	Remote Register #	Remote Device	Name	
1	28	Holding Register	UINT-16	2	ModSim	Data Value 2	
2	0	None	None	0	None		

The data direction is reversed but the same selections are still made. Select the local register that will be the source of data to write to the remote Modbus TCP device. Select the register type, data format, and register number to be written to in that device. Select a TCP server device from the list to write to. Only devices entered on the Devices page will appear here. If a unit number other than the default unit entered for this TCP server on the Devices page should be used, enter that unit number here. Click on the map number in the first column to access additional optional configuration parameters.

Map # 1 Update < Prev Next >

Read local register # 28 named Data Value 2

Write remote register when local register changes by > 0.000000 or when 0.0 seconds have elapsed with no change.

Otherwise write remote register unconditionally, applying local register data as follows:

Apply scale: 0.000000 and offset: 0.000000 Then if applicable, apply bit mask: 0000 and bit fill: 0000

Write Holding Register as Unsigned 16-bit Size: 0 with blank padding if checked

To register # 2 at ModSim unit # 1 With low register first if checked:

Repeat this process at least no more than every 5.0 seconds.

Enable this map only when index register 0 is set to a value of 0

Client Write Maps Enabled: 2 Insert Delete

The local register data may be written to the remote device periodically, or when it changes, or both. To send upon change (send on delta), check the first box and enter the amount by which the local register must change before being written to the remote device. To guarantee that the remote register will be written at least occasionally even if the data does not change, check the second box and enter some amount of time. This time period will be referred to as the "maximum quiet time".

Data from the local register may be manipulated before being written to the remote register. The local data is first multiplied by the scale factor. The offset is then added to it. If a bit mask is entered, and the remote register type is signed or unsigned integer (16-bit or 32-bit data), the mask will be bit-wise logical AND-ed with the data. The mask is right justified, then AND-ed with the data. The result is then left shifted back to the original position of the mask. In other words, the least significant bits of the original data will be stuffed at the position marked by the mask.

After the scaling and masking, the bit fill will be logically OR-ed into the result, but only if the mask was nonzero and was used. Both mask and fill are entered in hexadecimal. The effect of "fill" is that certain bits will always be set to 1 in the data written to the remote Modbus device.

Multiple local registers may be packed into a single remote register. To accomplish this, define two or more maps in sequence with the same remote destination. If the destination is the same, data types are 16 or 32-bit integer (signed or unsigned), bit masks are nonzero, and the maps are sequential, the results of all qualifying maps will be OR-ed together before being sent to the remote destination.

For the remote register to be written, select the register type, format, number, select a remote TCP device from the list, and enter a unit number if the default unit number for that device should not be used. Data formats are the same as described above for Read Maps. Size is only specified for character strings. Use INT-16 or UINT-16 data format for coils - in this case format only affects local register data conversion.

Modbus protocol treats all holding registers as strictly 16-bit registers. To

accommodate 32-bit or longer data, Modbus devices use multiple consecutive "registers" to hold the data. There is no standardization of whether the least significant part of the data comes first or last. Therefore, ValuPoint lets you set that according to whatever the remote Modbus device requires. If the least significant data is found in the first (or lower numbered) register in your Modbus device, then check the box after "With low register first".

The repeat time may determine how often the remote register will be written. If send on delta and maximum quiet time are not checked above, clicking the "at least" button will establish a periodic update time. If send on delta is used and you wish to limit the network traffic in the event changes are frequent, click the "no more than" button and enter the minimum time that should elapse before another write to the remote device. It is valid to select "no more than every 0.0 seconds" if you want all changes to be sent, but no periodic writes.

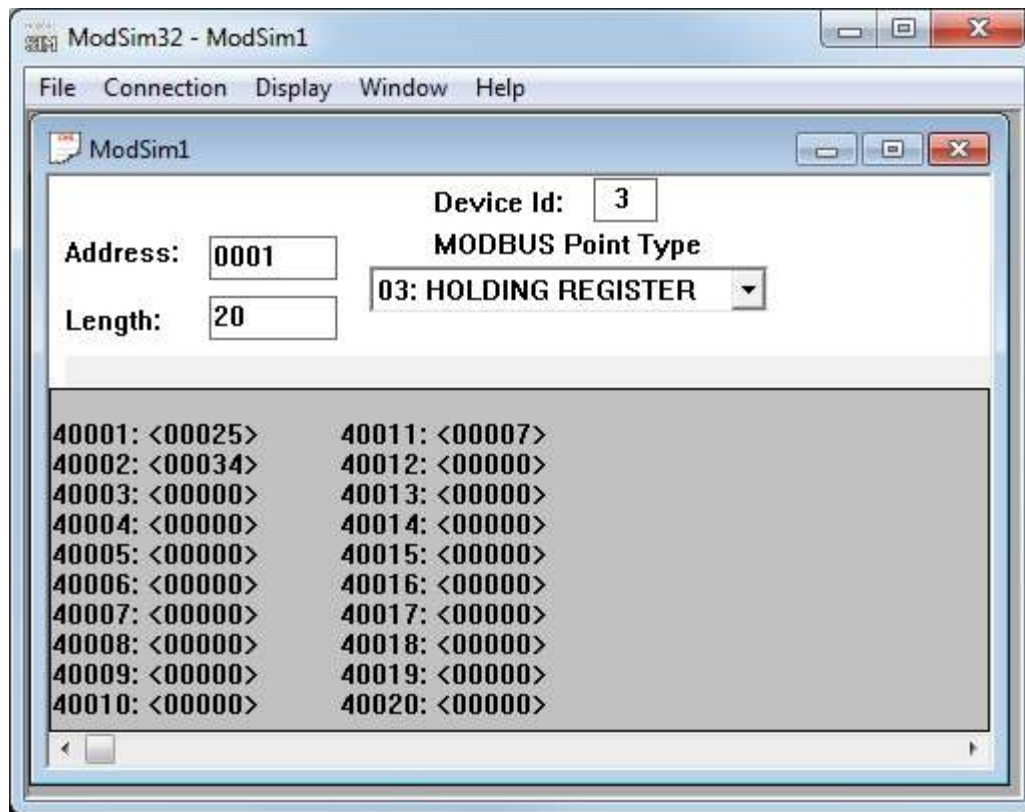
You have the option of making this Write Map conditional. If an index register number is provided and the Enable box is checked, then this write map will only be executed when the index register (local register) contains the value given. This allows multiple write maps to supply data to the same remote register based on the value of the local index register. It also allows writing to simply be suspended if a single write map supplies data to the remote register. In a more sophisticated scenario, you could potentially suspend writing of the remote device if you know the device is powered down.

Delete will remove the map number shown in the "Map #" box. Insert will insert a new map before the map number shown, and is used for placing maps between existing maps. It is not necessary to use Insert to add maps to the bottom of the list or to define any map presently having zero for a source register or "none" for remote type.

Selecting "none" for remote type effectively deletes the map even though it will still appear in the list until deleted. Unused maps at the end of the list will always show none as the type. If you wish to prevent these from being displayed, reduce the number of maps enabled.

Local Registers		Calculate	Copy	Report	
Showing registers from				27	Update < Prev Next >
Local Register #	Register Name	Set	Register Data	Register Format	
00027	Data Value 1	<input type="checkbox"/>	77	Unsigned 16-bit	
00028	Data Value 2	<input type="checkbox"/>	34	Unsigned 16-bit	
00029	Data Value 3	<input type="checkbox"/>	0	Unsigned 16-bit	

If the local register and remote register are not the same format, then data is converted automatically when written. In the example above, the Write Map is writing register 3, a floating point value, to remote register 2, an unsigned 16-bit value. The data is converted and rounded up, as illustrated below by ModSim acting as our Modbus TCP server here.



6.4 Modbus TCP Client Data Displayed by Server

The TCP Registers page shows a list of local registers mapped to TCP server devices. The page will show only one device at a time, and may have no entries if there are no maps for that device.



Click the Next Dev or Prev Dev buttons to go to the next or previous TCP device, or simply enter a number in the TCP Device # window and click Update. Registers for that server will now be displayed. In addition to a summary of the map (both read and write maps are shown), the time since last update is displayed. This time should generally be less than the poll time. If the last update time is large, it may mean there is an error preventing the update.

RTU Setup		RTU Data		TCP Setup		TCP Data	
TCP Registers		Error Counts		Errors: Read Maps		Errors: Write Maps	
TCP Device #1				Showing 1 to 2 of 2		Update < Prev Next >	
Dir.	Reg. Type	Remote Reg. #	Register Name	Local Reg. #	Update	Register Data	Time since Last update
From	Holding Reg	00001	Data Value 1	00027	<input type="checkbox"/>	77	2.000
To	Holding Reg	00002	Data Value 2	00028	<input type="checkbox"/>	34	1.000
TCP Device # 1		< Prev Dev		Next Dev >			

6.5 Modbus TCP Errors

The Error Counts page shows a tabulation, by TCP device, of all errors observed. In the example below, we can see that TCP device #1 is apparently not configured correctly as it is getting as many exception errors as messages sent.

Local Data		Modbus		IoT Cloud		System	
RTU Setup		RTU Data		TCP Setup		TCP Data	
TCP Registers		Error Counts		Errors: Read Maps		Errors: Write Maps	
Update							
Device	Reset	Total Messages	No Responses	Exceptions			
1	<input type="checkbox"/>	1146	0	1146			
2	<input type="checkbox"/>	0	0	0			
3	<input type="checkbox"/>	0	0	0			

If the counts show some problems, we can look for more detail on the Errors: Read Maps (or Errors: Write Maps) pages. These pages will tell us exactly which Read Map (or Write Map) the problem is occurring on, and what the error is, as illustrated below.

TCP Registers		Error Counts		Errors: Read Maps		Errors: Write Maps	
Update							
Map #	Register Name	Error Description		Exception Code			
1	Data Value 1	Exception code returned by device		Illegal data address			

If you see total messages of zero and a "no responses" count greater than zero, it means the ValuPoint was not able to connect to the IP address of the TCP server. Without being able to connect at all, there was never an attempt to send a message, and hence zero total messages while "no responses" continues to increment.

TCP Registers		Error Counts	Errors: Read Maps	Errors: Write Maps	
<input type="button" value="Update"/>					
Device	Reset	Total Messages	No Responses	Exceptions	
1	<input type="checkbox"/>	0	4	0	
2	<input type="checkbox"/>	0	0	0	

When "no responses" is indicated, the Errors: Read Maps (or Write Maps as applicable) page will show that the response timed out, but this is typically a foregone conclusion when you see the "no responses" count for a TCP device.

TCP Registers		Error Counts	Errors: Read Maps	Errors: Write Maps	
<input type="button" value="Update"/>					
Map #	Register Name	Error Description		Exception Code	
1	Data Value 1	Response timed out		---	

When you get any type of connection related problem with a TCP device, the connection status will typically give you some clues.

Devices	Client Read Map	Client Write Map		
Device # <input type="text" value="1"/>	<input type="button" value="Update"/> <input type="button" value=" < Prev"/> <input type="button" value=" Next >"/>			
Local Name <input type="text" value="Device 1"/>				
Use <input checked="" type="radio"/> Static IPv4 <input type="radio"/> Static IPv6 <input type="radio"/> Domain Lookup				
IP Address <input type="text" value="192.168.1.134"/>	Port: <input type="text" value="502"/>			
Domain Name <input type="text"/>				
Unit (optional) <input type="text" value="1"/> <input type="checkbox"/> Use FC 5/6 instead of 15/16				
Default Poll Period <input type="text" value="5.0"/> Seconds	Connection Status <input type="text" value="118"/>	<input type="button" value=" Clear"/>		

Connection status codes you may see include:

5 = Connection attempt timed out, unable to establish connection (usually means remote device not connected or not reachable)

104 = Connection reset by peer

111 = Connection refused

113 = Connection aborted

114 = Network is unreachable

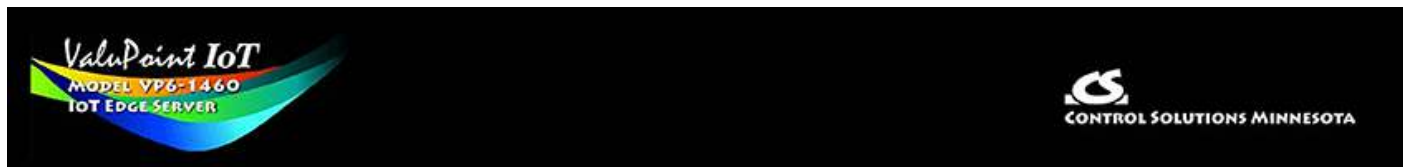
115 = Network interface not configured

116 = Connection timed out

118 = Host is unreachable

125 = Address not available

205 = DNS error



7. Configuring ValuPoint as a Modbus RTU Slave

7.1 Modbus RTU Device Configuration

Device configuration for RTU means configuring the serial port. Select the baud rate and parity as applicable. Select "I am a Slave". It is important to provide an address or unit number that is not used by any other slave on the RTU network. The poll rate, timeout, and "Use FC 5/6..." only apply when RTU is master.

The screenshot shows a configuration window with a dark green background. At the top, there are tabs for 'RTU Setup', 'RTU Data', 'TCP Setup', and 'TCP Data'. Below these are sub-tabs for 'Local Device', 'RTU Read Map', and 'RTU Write Map'. An 'Update' button is in the top right. The main area has two sections: 'I am the Master' (disabled) and 'I am a Slave' (selected). Under 'I am the Master', there are fields for 'Default Poll Rate' (0.000) and 'Timeout' (0.000), both in seconds. Under 'I am a Slave', there is a field for 'My Address or Unit #' (1). At the bottom, there is a checkbox for 'Use FC 5/6 instead of 15/16 for unit numbers (slave addresses) starting at' followed by a field containing '0'. The 'Baud Rate' is set to 19200 and 'Parity' is set to 'None, 1 Stop Bit'.

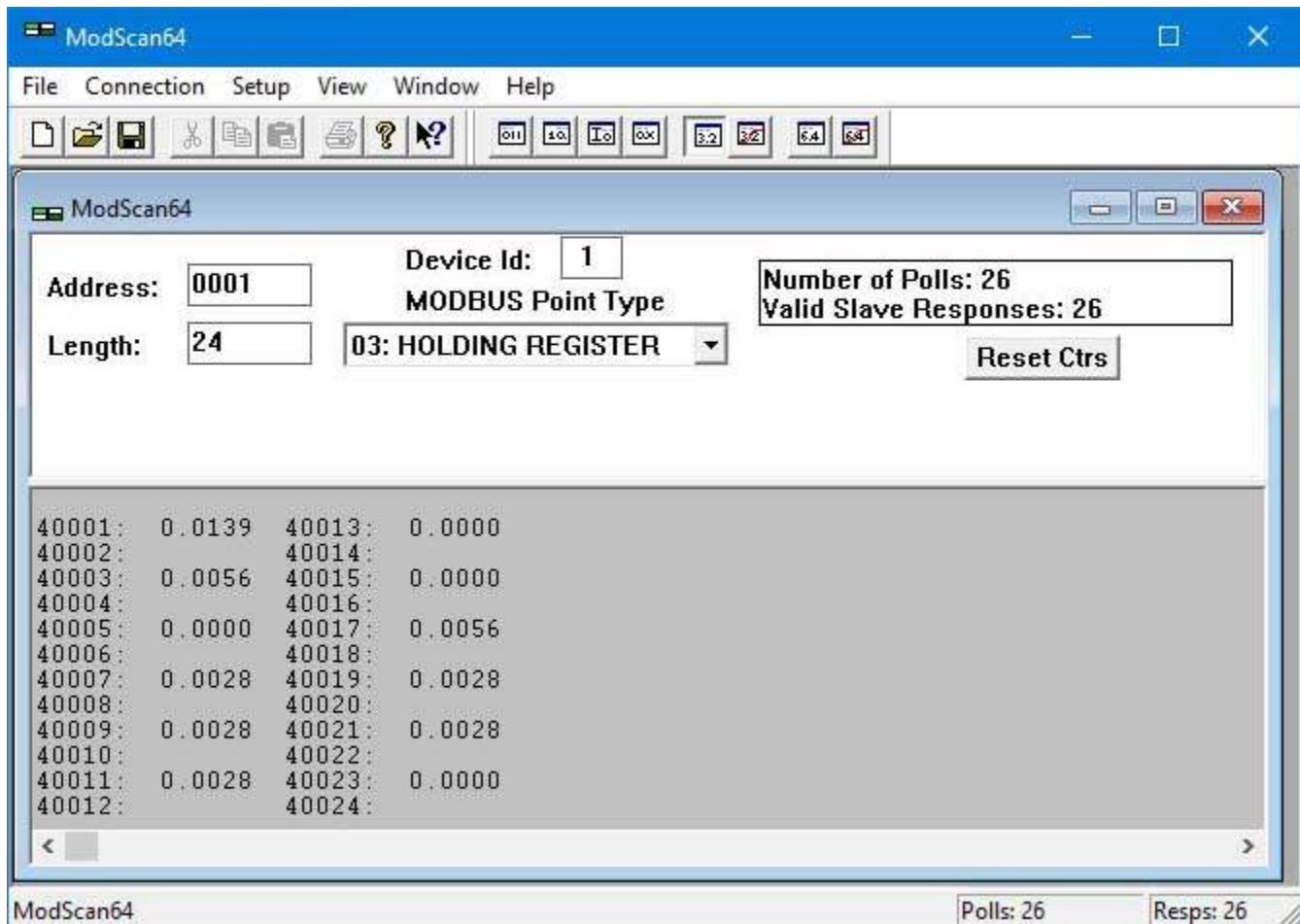
7.2 Modbus RTU Slave Register Map

The local registers in the ValuPoint will be most often accessed as holding registers, but can also be accessed as input registers (for reading but input registers cannot be written to). If the local register is defined as 16-bit signed or unsigned, then it can also be accessed as a coil or discrete input (for reading). When accessed as a single bit Modbus register, the value read by the Modbus master will be 0, or 1 if the local register contains 1 or any other non-zero value. Of course the remote master can only write 0 or 1 to a coil. Note also that a local register defined as something bigger than 16-bit cannot be accessed as a coil or discrete input.

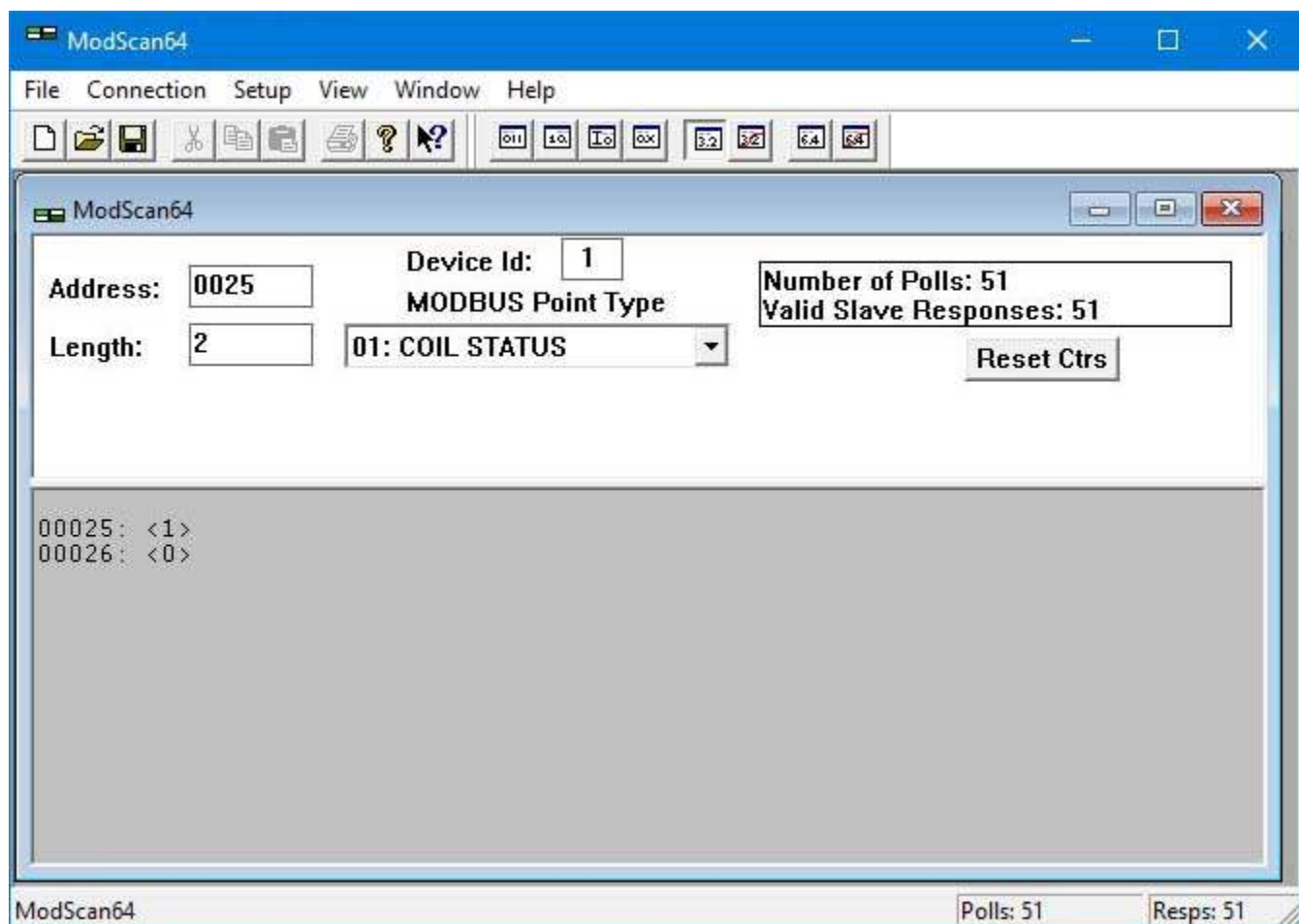
The register numbers that the remote Modbus RTU master should read or write are simply those shown in the first column on the Local Registers page.

Local Data		Modbus		IoT Cloud		System	
Data			Events				
Local Registers		Calculate		Copy		Report	
Showing registers from				1	Update		< Prev Next >
Local Register #	Register Name	Set	Register Data		Register Format		
00001	A/UI #1	<input type="checkbox"/>	0.027847		Single Float		
00003	A/UI #2	<input type="checkbox"/>	0.000000		Single Float		
00005	A/UI #3	<input type="checkbox"/>	0.002785		Single Float		
00007	A/UI #4	<input type="checkbox"/>	0.002785		Single Float		
00009	A/UI #5	<input type="checkbox"/>	0.000000		Single Float		
00011	A/UI #6	<input type="checkbox"/>	0.005569		Single Float		
00013	A/UI #7	<input type="checkbox"/>	0.002785		Single Float		
00015	A/UI #8	<input type="checkbox"/>	0.000000		Single Float		
00017	A/UI #9	<input type="checkbox"/>	0.002785		Single Float		
00019	A/UI #10	<input type="checkbox"/>	0.000000		Single Float		
00021	A/UI #11	<input type="checkbox"/>	0.000000		Single Float		
00023	A/UI #12	<input type="checkbox"/>	0.005569		Single Float		
00025	DO #1	<input type="checkbox"/>	1		Unsigned 16-bit		
00026	DO #2	<input type="checkbox"/>	0		Unsigned 16-bit		

The first 24 registers are defined as floating point register pairs and assigned to the 12 physical input points. Viewing the floating point data using ModScan is illustrated below. The default data format is "Most significant register first" when selecting floating point in ModScan.

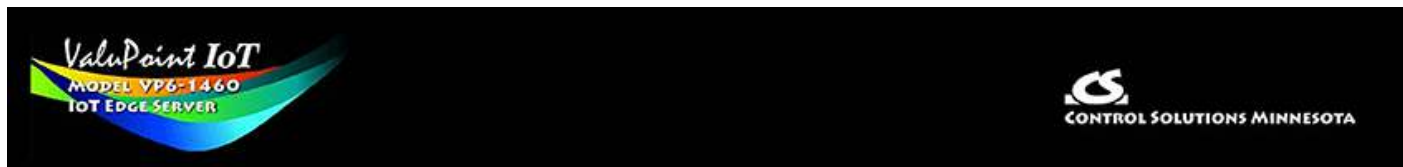


The two registers assigned to the relay outputs, registers 25 and 26, may be read and written as coils.



7.3 Modbus RTU Slave Diagnostic

The ValuPoint brand new out of the box will be configured as Modbus RTU slave, 9600 baud, slave address 1. Although no registers other than I/O points are configured, there will be a single holding register accessible for diagnostic purposes, at register number 8801 (or 48801 if using Modicon notation). The content of this register will be firmware revision expressed as a 3-digit number "abbcc" where "a" is the major revision, "bb" is minor revision, and "cc" is build iteration. This should correspond to the firmware revision displayed on the home page (index.html) of the web user interface for the device, which is displayed as "a.bb.c".



8. Configuring ValuPoint as a Modbus TCP Server

8.1 Modbus TCP Device Configuration

There is really little to do to configure the ValuPoint to be a Modbus TCP server. The gateway needs an IP address and you have already set that via the Network page. The only other thing is to verify that the Modbus Port number is set to a non-zero number. Port 502 is the port set aside for standard Modbus TCP use and should be used unless you have a specific reason not to.

If you will not be using Modbus TCP and wish to disable it, enter zero for Modbus Port, and click Set Ports. Following the next restart, you will be unable to connect via Modbus TCP with port set to zero.

IMPORTANT: The Modbus port will be initially set to zero as shipped from the factory. You will need to change it to 502 and restart before connecting via Modbus TCP for the first time.

The screenshot shows the 'Network' configuration page in the ValuPoint web interface. The page has a dark green header with tabs for 'File Manager', 'Network', 'Resources', 'User', 'I/O Config', and 'User'. The 'Network' tab is active. The main content area is dark blue and contains the following settings:

- Web Server: HTTPS Enabled (on 443) HTTP Enabled
- HTTP Port: (default 80)
- Modbus Port: (default 502)
- FTP Server: Enabled
- REST API: Enabled
- MAC Address: 00:40:9D:DC:0D:DD
- System Uptime: 8,00:08:06
- HTTPS certificate status: Ok

8.2 Modbus TCP Server Register Map

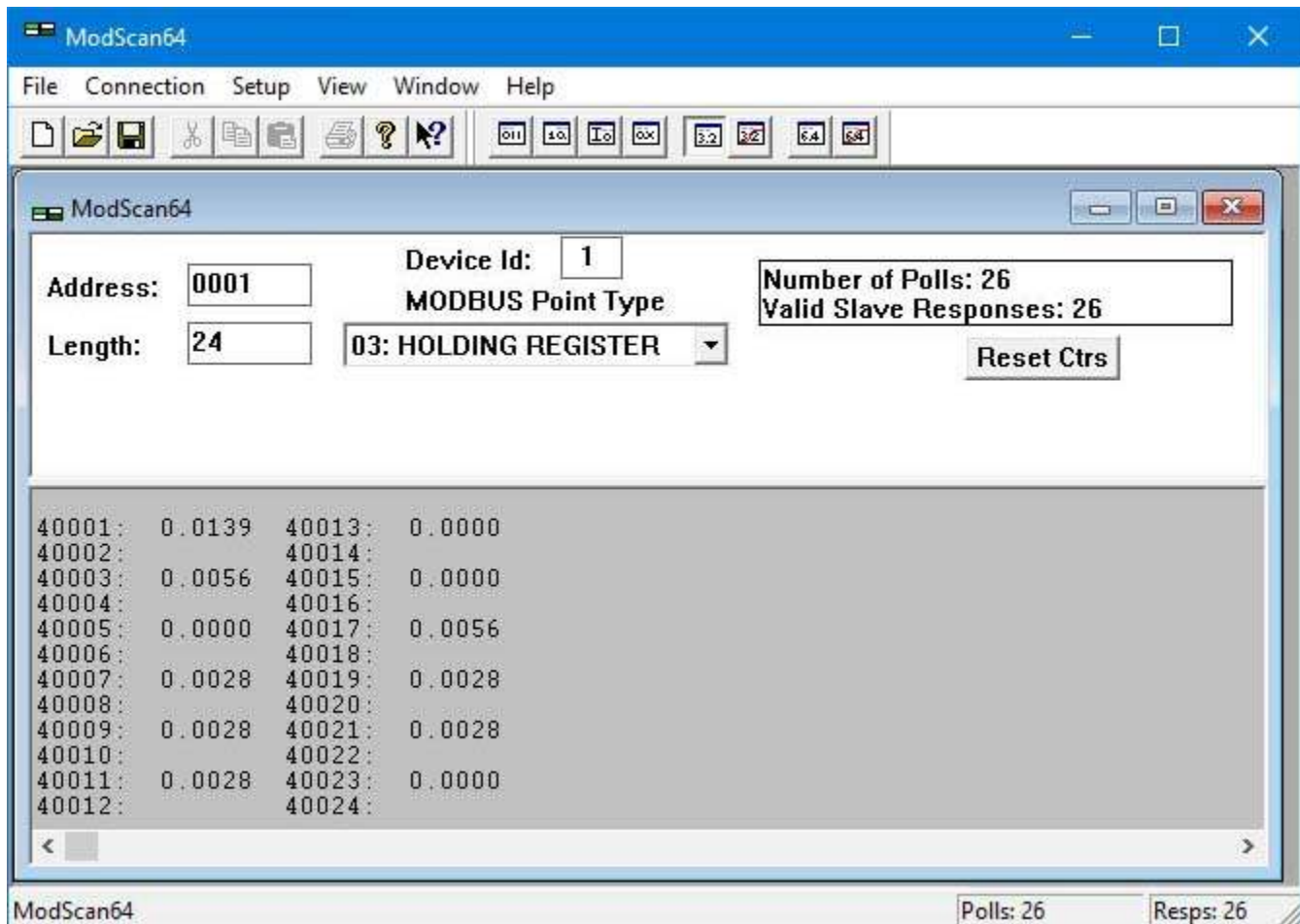
The local registers in the ValuPoint will be most often accessed as holding registers, but can also be accessed as input registers (for reading but input registers cannot be written to). If the local register is defined as 16-bit signed or unsigned, then it can also be accessed as a coil or discrete input (for reading). When accessed as a single bit Modbus register, the value read by the Modbus master will be 0, or 1 if the local register contains 1 or any other non-zero value. Of course the remote master can only write 0 or 1 to a coil. Note also that a local register defined as something bigger than 16-bit cannot be accessed as a coil or discrete input.

The register numbers that the remote Modbus TCP client should read or write are

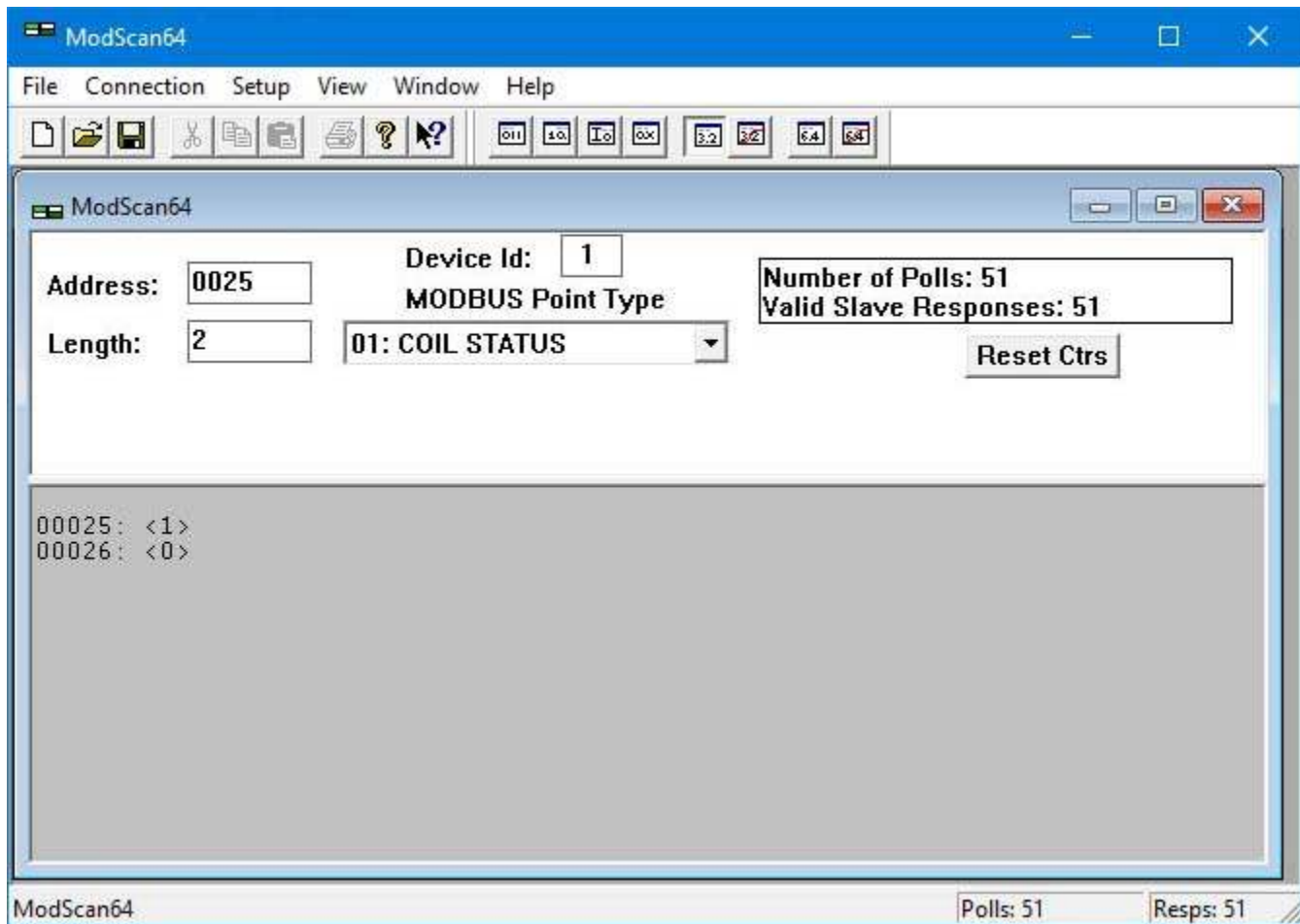
simply those shown in the first column on the Local Registers page.

Local Data		Modbus		IoT Cloud		System			
Data			Events						
Local Registers		Calculate		Copy		Report			
Showing registers from						1	Update	< Prev	Next >
Local Register #	Register Name	Set	Register Data				Register Format		
00001	A/UI #1	<input type="checkbox"/>	0.027847				Single Float		
00003	A/UI #2	<input type="checkbox"/>	0.000000				Single Float		
00005	A/UI #3	<input type="checkbox"/>	0.002785				Single Float		
00007	A/UI #4	<input type="checkbox"/>	0.002785				Single Float		
00009	A/UI #5	<input type="checkbox"/>	0.000000				Single Float		
00011	A/UI #6	<input type="checkbox"/>	0.005569				Single Float		
00013	A/UI #7	<input type="checkbox"/>	0.002785				Single Float		
00015	A/UI #8	<input type="checkbox"/>	0.000000				Single Float		
00017	A/UI #9	<input type="checkbox"/>	0.002785				Single Float		
00019	A/UI #10	<input type="checkbox"/>	0.000000				Single Float		
00021	A/UI #11	<input type="checkbox"/>	0.000000				Single Float		
00023	A/UI #12	<input type="checkbox"/>	0.005569				Single Float		
00025	DO #1	<input type="checkbox"/>	1				Unsigned 16-bit		
00026	DO #2	<input type="checkbox"/>	0				Unsigned 16-bit		

The first 24 registers are defined as floating point register pairs and assigned to the 12 physical input points. Viewing the floating point data using ModScan is illustrated below. The default data format is "Most significant register first" when selecting floating point in ModScan.

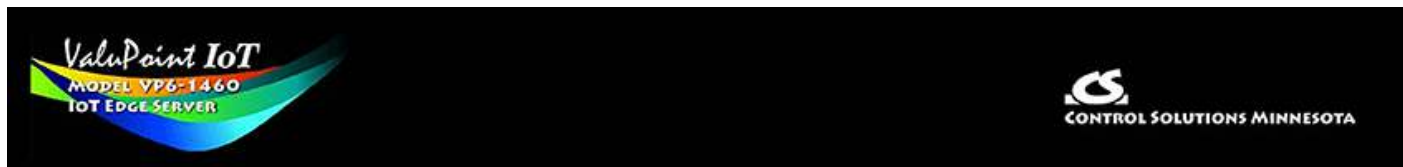


The two registers assigned to the relay outputs, registers 25 and 26, may be read and written as coils.



8.3 Modbus TCP Server Diagnostic

The ValuPoint brand new out of the box will have no registers configured other than I/O points. There will be a single holding register accessible for diagnostic purposes, at register number 8801 (or 48801 if using Modicon notation). The content of this register will be firmware revision expressed as a 3-digit number "abbcc" where "a" is the major revision, "bb" is minor revision, and "cc" is build iteration. This should correspond to the firmware revision displayed on the home page (index.html) of the web user interface for the device, which is displayed as "a.bb.c".



9. Configuring Event Rules

Alarm monitoring is the most common use for an event rule, but event rules can also be incorporated into control algorithms to cause some action to happen as the result of some given condition. When used for alarm monitoring, the event rule can result in automatically sending an email notification to your smart phone or computer.

9.1 Event Rule List

The Event Rules page displays a list of currently defined event rules in summary form. Click on the rule number in the first column to see and modify the full rule.

Rule #	Event Name	Local Register #	Register Data	Test Criteria	Test Value	State*	Error Code
<u>1</u>	Test Event 1	00027	0	greater than	10.00000	False	0
<u>2</u>	Test Event 2	00028	0	less than	10.00000	True	0
<u>3</u>	Test Event 3	00029	0	equal to	10.00000	False	0
<u>4</u>	Test Event 4	00030	0	greater or equal to	10.00000	False	0
<u>5</u>	Test Event 5	00031	0	less or equal to	10.00000	True	0
<u>6</u>	Test Event 6	00032	0	not equal to	10.00000	True	0
<u>7</u>	Test Event 7	00033	0	changed by	5.000000	False	0
<u>8</u>	Test Event 8	00034	0	increased by	5.000000	False	0
<u>9</u>	Test Event 9	00035	0	decreased by	5.000000	False	0
<u>10</u>	Test Event 10	00036	0	deviates from	10.00000	True	0
<u>11</u>		00000		None	0.00	False	0

Use Next and Prev to scroll through the list if there are many events. Click Update to see a refresh of current state. Enter a number in the Showing window and click Update to jump to that point in the table.

9.2 Event Rule Details

The full event rule looks like this, and the various parts of it are explained in detail below.

The screenshot displays the 'Event Rules' configuration page. At the top, there are tabs for 'Event Rules', 'Data Logging', 'Templates', and 'Recipients'. The 'Event Rules' tab is active, showing a rule configuration for 'Test Event 1'. The rule number is 1, and it is currently disabled ('Rule presently tests False'). The configuration includes:

- Read local source register # 27 for this event named Test Event 1
- Event is TRUE if the value is greater than 10.000000
- Qualified by this hysteresis value: 0.000000
- Set local destination register # 0 as follows below while logging on-time to register # 0
- If true, to a value which is this value: 0.000000
- If false, to a value which is this value: 0.000000
- Email to user group 1 using template # 1 upon transition true, no more than every 0 minutes
- Email to user group 1 using template # 1 upon transition false, no more than every 0 minutes
- # Rules Enabled: 11

 Buttons for 'Update', '< Prev', and 'Next >' are visible at the top right. 'Insert' and 'Delete' buttons are at the bottom right.

The number of rules enabled simply limits the scope of display on the tabular event rule list. To scroll through from one event to the next on the event detail page, use Next and Prev. To jump to a different rule number, enter it in the Rule # window at the top and click Update. Insert will insert a new blank rule before the currently displayed rule. Delete will remove the currently displayed rule.

This close-up screenshot shows the top portion of the event rule configuration. It displays 'Read local source register # 27 for this event named Test Event 1'. Below this, it shows the test criteria: 'Event is TRUE if the value is greater than 10.000000' and 'this local register: 0'.

An "event" occurs when the value contained in a register meets some criteria that you have specified on this page. Start by selecting the register number that this test will be applied to. Give the event a name. In addition to being a reference for documentation purposes, this event name may be included in email messages generated by this event.

Select a test type, such as greater than, from the test list. Provide a threshold. In the above example, the event is "true" when register 1 contains a value greater than 10, and if email is configured, the sending of an email would be triggered upon register 1 crossing this threshold.

If you would like to have the threshold set through some other register so that it can be readily changed on the fly, select local register instead and provide that register number from which the threshold will be taken each time the rule is evaluated. Event rules are re-evaluated several times per second.

The screenshot shows the configuration interface for an event rule. A dropdown menu is open, listing comparison operators: greater than, less than, equal to, greater or equal to, less or equal to, not equal to, changed by, increased by, decreased by, and deviates from. The 'greater than' option is selected. The interface includes various input fields for values, local registers, and hysteresis, as well as checkboxes for email notifications.

The possible test types are illustrated above. Some tests need further explanation. The "Changed by" test means amount of change since the last event transition to true. If the local register has changed by the value specified as "this value" or the value contained in the local register referenced, the test is true. The "Changed by" value can be an increase or decrease. To consider the event to be true only upon increase or decrease since the last transition, select those tests instead. The "Deviates by" uses a special application of the hysteresis value. If the present value of the local register deviates from the threshold by the margin set as hysteresis, then this test will be deemed to be "true". This amounts to a combined greater than and less than in the same test.

IMPORTANT: When using any of the "change" tests, and using email notifications, you should **ONLY** select email upon transition to true because the change will only be true for one instant and then the comparison threshold is moved and the rule immediately becomes false again. The result, if you enable email on both true and false, is that you will always get 2 emails right away each time there is an incremental change.

This screenshot shows the configuration for hysteresis, on-time, and off-time. The hysteresis value is set to 0.000000, the minimum on-time is 0:00:00, and the minimum off-time is 0:00:00.

Qualifications are optional, and enabled only when values are nonzero. How hysteresis is applied depends on the comparison. For a test that becomes true if greater than, the test will not return to false until the local register is less than the test value by a margin of at least this hysteresis value. If a test becomes true if less than, it will not return to false until the local register is greater than the test value by a margin of at least this hysteresis value.

On time and off time, if specified, determine how long the condition must be true (on time) or false (off time) before the true or false response is actually taken. Times are given in HH:MM:SS format (hours, minutes, seconds). If the condition goes away before this time is up, then it will be as if the event never happened in the first place.

This screenshot shows the configuration for the local destination register and the response. The local destination register is set to 0. The response is configured to be the same as the source, with a value of 0.000000, and to be from the local register # 0.

Now that you have specified what the condition is, you proceed to define the response. Start by selecting which local register the response is applied to. This will be known as your destination register. Typically this register will be linked to an output. **IMPORTANT**

note for email notifications: You do not need to apply the result to any destination register if you simply want to report the event via an email message. Leave the destination register set to zero and ignore the 2 lines that follow it. The result of the test will be processed as true or false by email notification processing without any destination register specified.

The first line after the destination register number is the response that will be taken when the condition is true, and the following line is the response that will be taken when the condition is false. Either the source register is copied, a fixed value is applied, or another register is used to provide the data written to the destination register.

The "on-time" logging is optional, and may be used without setting any destination register. It simply records the amount of time the threshold rule tests true, and records that time in the register given (if nonzero). Time is recorded in minutes. If the register is an integer register, you may record up to 65,000 minutes (or 32,000 minutes if treated as Signed integer). Much longer times may be recorded if a floating point register is used. The logged time may be reset by simply writing 0 to the register via the web page or via Modbus.

Follow above rule only if local register is set to a value of

You have the option of enabling processing of this rule only when a selected local register contains a given value. Any local register may be used as the enable register. If the event had previously transitioned to true when the enable register changes to a disable value, then the rule will be processed as a transition to false one time.

Email to user group using template # upon transition true, no more than every minutes

Email to user group using template # upon transition false, no more than every minutes

Email to user group using template # every hours (true or false)

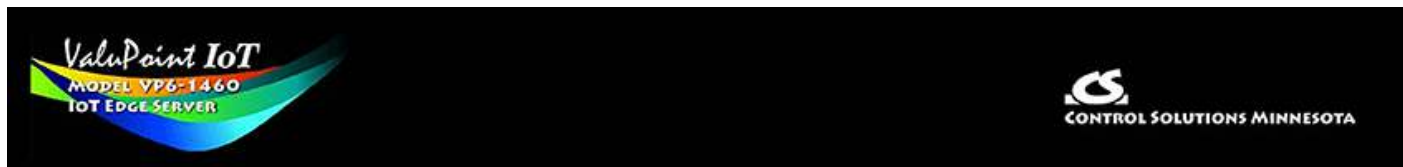
Time since last email (HH:MM:SS) **0:00:20** Email type: Transition to True, Email error code: 0

Email notifications will be generated as given if they are enabled. Emails are not required - event rules can be used to simply set register values internally without any email notifications. You can use hysteresis and minimum on/off times to minimize spurious transitions, but you may also limit the frequency of email notifications using the "no more than" time limit. If "no more than" is zero minutes, then there is no limit. Think carefully about whether conditions could exist that will flood your email inbox.

If you use email notifications, you have the option of sending an email when the rule transitions to true, or transitions to false, or both. You also have the option of periodically sending an email regardless of condition. If you are sending an email about an alarm condition that doesn't happen very often, you may want to configure a daily email that gets sent regardless of state just to tell you that your monitoring device is still there.

The "group" refers to a user group that was set up on the Recipients page. The "template #" refers to a template that was set up on the Templates page. If email

error code is anything other than zero, there was a problem, and the codes are explained on the Recipients page where you may also send a test email.



10. Configuring Email Client & Notifications

Email messages are sent if desired for event notifications and also for data log file delivery. The email message is constructed from templates you create. They are sent to email addresses you provide. The emails are sent via an email server or account you provide. The email account used here can be a Gmail account.

Once email has been set up, there are three places where emails are actually sent: (1) Event Rules, (2) Data Logging, (3) Test email from Recipients page pictured later in this section.

10.1 Assigning Email Templates

Any email message generated by this device is created from a template that you create and then assign to a template number.

Template #	File Name
1	fulllog.txt
2	templateAll1.txt

Manage Template Files Selected file: /FS/FLASH0/templateAll1.txt

File Directory:

New File: To Template #

The template is a simple text file with a .txt suffix and could be edited externally and uploaded via the File Manager page. It can also be edited here.

There are two steps to using an email template: (1) Create the template, (2) Assign that template file to a template number that can be referenced in event rules or data logging setup. Currently assigned template files are displayed in the list.

To create a new template here, enter a file name ending in ".txt", then click the New button. Next, click the Edit button and refer to editing the template below.

Once you have created a template that shows up in the File Directory list (only *.txt files will be displayed in this list), select a file from the list (pick file from drop-down list, then click Select), enter a number in the "To Template #" window, and then click Assign. The template file itself is saved in the Flash file system when you Save it or upload it. However, the assignment of a file name to a template number is part of the configuration you save as an XML file on the File Manager page. Don't forget to save your configuration after assigning a template.

To View a file, you simply need to pick the file name in the drop-down list and click View. To select for editing or assignment, you need to pick a file and then click Select. Upon clicking the Select button, that file name will show up in the File window.

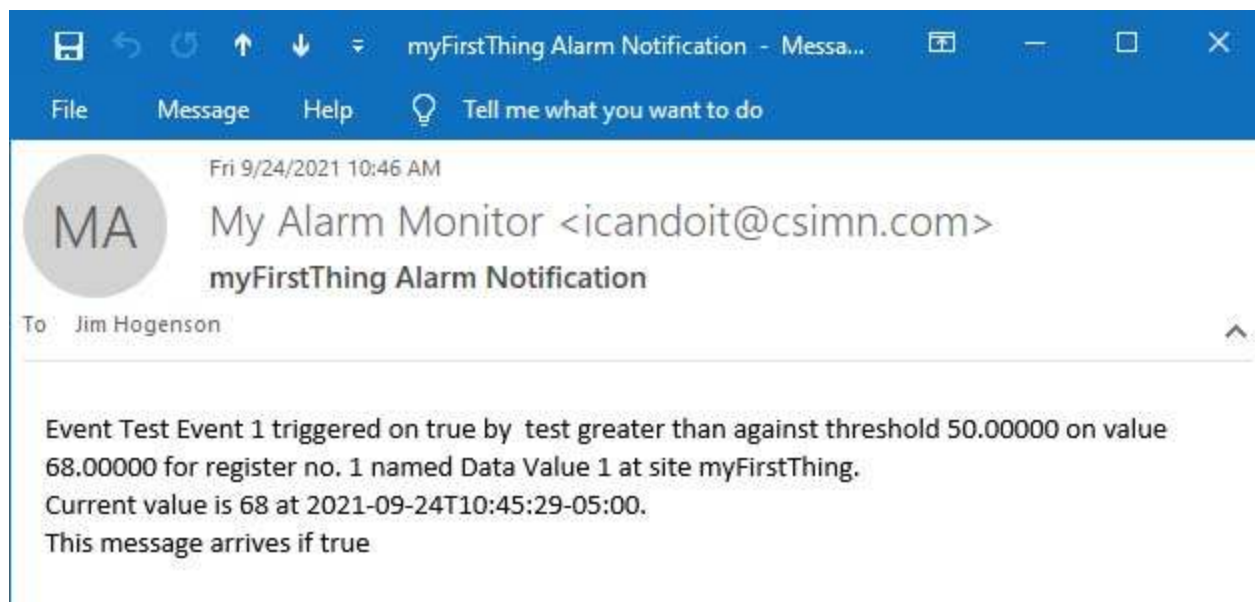
10.2 Editing Email Templates

The editing page will be blank when you first get here. If you just created a new file, it will be empty, so just go ahead and start typing. If you are editing an existing file, click Get to read the file. It is unlikely that your template will be so long it doesn't fit on this page, but if so, use the Page Down and Page Up buttons.

When you have finished editing your template, click Save to save it to the Flash file system.



The above example template produced the following email.



The first line of the template file should start with SUBJECT: if the email should have a subject. If this line is omitted (or placed elsewhere), there will be no subject on the email.

The remainder of the template will be copied verbatim, except for those variable names or tags enclosed in brackets. The variables will be replaced with real time data at the time the email is sent. Line breaks in the template are not copied. To get a line break in the message actually sent, include {eol}.

The example illustrated above includes all of the possible template variables that might be used in creating an email message. Available variables or tags are as follows:

{name}

Thing Name on the Thing ID page under IoT Cloud is used to identify the location when using the AWS Cloud. Regardless of whether using AWS, this Thing Name is also used as a location identifier for email notifications.

{regnum}

Local register number tested by event rule

{regname}

Local register name

{value}

Value looked up as of time message is sent

{eventvalue}

Value as of event rule transition

{testvalue}

Value the rule tested against (threshold)

{testtype}

Type of test (e.g. greater than)

`{state}`

Insert "true" or "false" state of event right now (applies to periodic reporting)

`{eventname}`

Name of event given in event rule configuration

`{timestamp}`

Timestamp as of when message sent

`{true:xxx yyy zzz}`

Literal string conditional, include "xxx yyy zzz" in message only if event is or transitioned to true

`{false:xxx yyy zzz}`

Literal string conditional, include "xxx yyy zzz" in message only if event is or transitioned to false

`{eol}`

Insert line break (breaks in template are not copied, only the `{eol}` tag results in a break in the message sent)

10.3 Email Recipients

The people to whom you wish to send emails are listed here. Each recipient can be a member of any or all of 5 "groups". When an event is configured to send an email notification, it will be designated to be sent to one of these groups. Thus an event can be sent to many recipients, and different events can be sent to different recipients.

User #	Email Address	Name	Group 1	Group 2	Group 3	Group 4	Group 5
1	jimhogenson@csimn.com	Jim Hogenson	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Email Server Setup:

SMTP Host: csimn.com

SMTP Port: 465

User Name/Email: icandoit@csimn.com

Password: ●●●●●●●●

"From" Name: Full Log Test

Send Test Email to Group: 0 Using Template = 0 Email Error Code: 0 Refresh

10.4 Email Server

This is where you enter the host information and account credentials for the email

server you will use to send emails. You can use an IP address for SMTP Host. If you use a host name, be sure you have also configured a DNS server on the Network setup page. Provide the port number applicable to your host. The host and port that you would use for Gmail is illustrated.

Create a Gmail account if you don't already have one to use for this purpose (or use whatever other account you like). Provide the user name and password that will allow this IoT device to log in. The "From" name can be anything - it is what shows up as the "From" name in the email. Click Update Server, and then to retain these settings, go to the File Manager page and save your configuration.

User #	Email Address	Name	Group 1	Group 2	Group 3	Group 4	Group 5
1	jimhogenson@csimn.com	Jim Hogenson	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Email Server Setup:

SMTP Host:

SMTP Port:

User Name/Email:

Password:

"From" Name:

Send Test Email to Group Using Template # Email Error Code: Refresh

You may send a test email to any of the five email groups you have configured using any template you have created and assigned. The test email will not have an actual event to reference, so any variables that would otherwise be event information will be filled in with dummy values.

If the test email is unsuccessful, a non-zero error code will be displayed here. There may be a delay between clicking Send and seeing the result, so click Refresh a little later to check the outcome.

Email error codes can be any of the following:

0 = No errors

+1 = No recipients match selected group number

-1 = Unable to allocate memory to build email message

-2 = No DNS server found

-3 = DNS could not find host

-4 = Server lookup attempt ran into other errors

-5 = Failure to create socket

-6 = Failed to handshake or negotiate a TLS connection with server

- 7 = Failed to authenticate with the given credentials
- 8 = Failed to send data to server
- 9 = Failed to receive data from server
- 10 = Failed to properly close connection
- 11 = SMTP server sent back an unexpected status code
- 12 = Invalid parameter
- 13 = Failed to open or read a local file
- 14 = Failed to get a local date and time

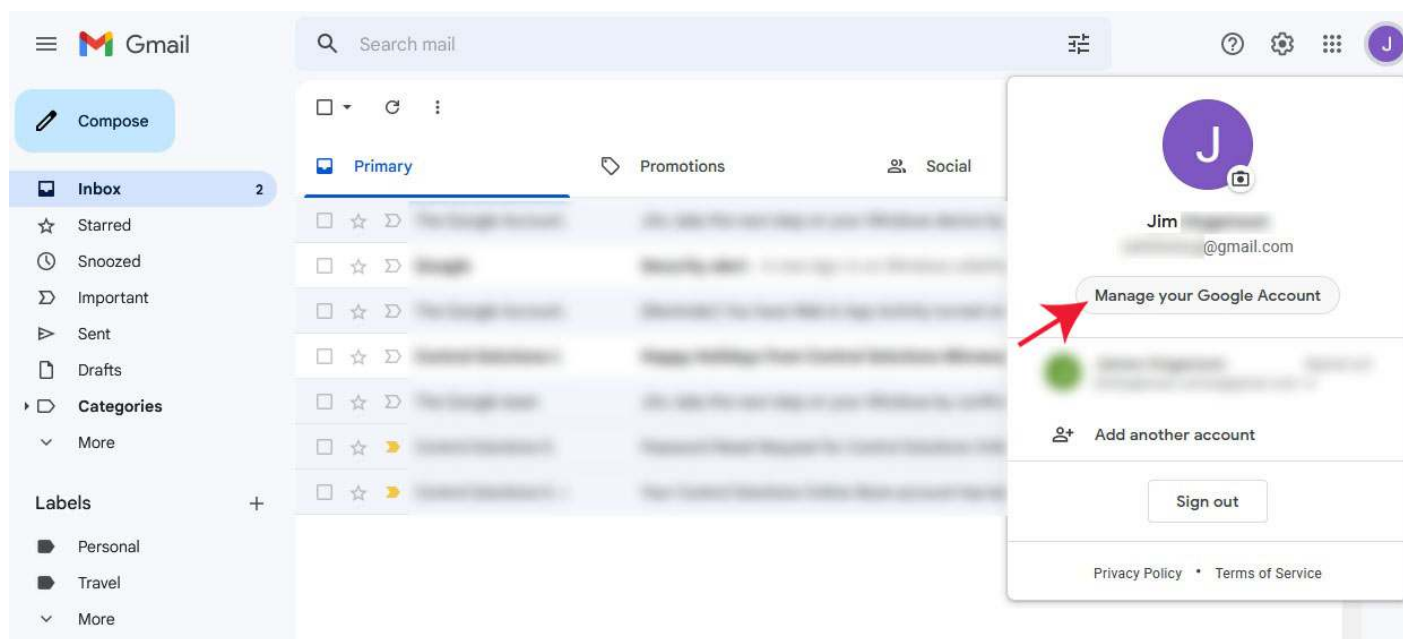
Error codes associated with emails sent by event rules are displayed on the respective event rule page. Error codes associated with emails sent by the data logger are displayed on the data logger page.

10.5 Using Gmail

As per notice from Google's Less Secure App Deprecation Notice, on May 30, 2022 the ability access Google accounts through 'less secure apps' is being deprecated. The preferred method is OAuth 2.0; however, this requires 2-factor authentication which your remote device is not going to handle well because it doesn't have a cell phone in its pocket.

The alternative is to go to your Google account and create an apps password, and once this is created, just use the apps password instead of the actual password for your gmail account.

Log into the Gmail account that you created for your device to use. Click the icon in the upper right corner, and then select Manage your Google Account.



Under account management, select Security.

Google Account

Search Google Account

Home

Personal info

Data & privacy

Security

People & sharing

Payments & subscriptions

About

Welcome, Jim

Manage your info, privacy, and security to make Google work better for you. [Learn more](#)

Privacy & personalization

See the data in your Google Account and choose what activity is saved to personalize your Google experience

[Manage your data & privacy](#)

You have security recommendations

Recommended actions found in the Security Checkup

[Protect your account](#)

Turn on 2-Step Verification if not already on. You will be asked for a verification code sent to your cell phone during this process. After verifying that 2-Step Verification is "On", click on the App passwords selection.

Google Account

Search Google Account

Home

Personal info

Data & privacy

Security

People & sharing

Payments & subscriptions

About

Signing in to Google

Password Last changed Sep 15, 2010

2-Step Verification On

App passwords 1 password

The screen shot below illustrates that an app password named MQ-61 has already been added. To add a password, under "Select app", choose Other, and enter a name for your app (e.g. MQ-61). Then under "Select device", choose Other and provide a custom name. Then click Generate.

← App passwords

App passwords let you sign in to your Google Account from apps on devices that don't support 2-Step Verification. You'll only need to enter it once so you don't need to remember it. [Learn more](#)

Your app passwords

Name	Created	Last used	
MQ-61	11:18 AM	11:21 AM	🗑️

Select the app and device you want to generate the app password for.

Select app

- Mail
- Calendar
- Contacts
- YouTube
- Other (Custom name)

Select device

- iPhone
- iPad
- BlackBerry
- Mac
- Windows Phone
- Windows Computer
- Other (Custom name)

GENERATE

Upon clicking Generate, a screen like the following will be displayed. Disregard "seuresally@gmail.com", that is just an example. Your app password is highlighted in the yellow box.

Google Account

← App passwords

App passwords let you sign in to your Google Account from apps on devices that don't support 2-Step Verification. You'll only need to enter it once so you don't need to remember it. [Learn more](#)

Your app passwords

Generated app password

Your app password for your device

larh obhs plvi knlh

How to use it

Go to the settings for your Google Account in the application or device you are trying to set up. Replace your password with the 16-character password shown above. Just like your normal password, this app password grants complete access to your Google Account. You won't need to remember it, so don't write it down or share it with anyone.

DONE

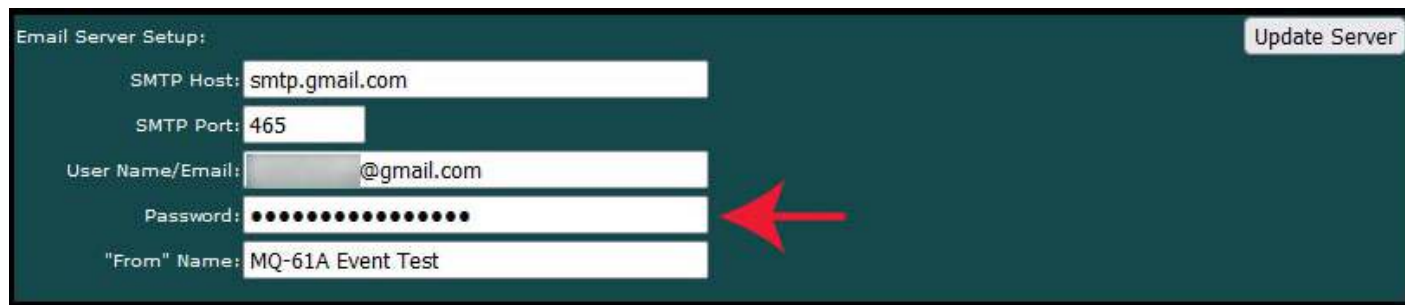
Email

seuresally@gmail.com

Password

●●●●●●●●

Copy from the yellow box by highlighting the text and then right-click "Copy". You will note that the spaces displayed will be removed. If you copy by typing the password, do not include the embedded spaces - they are just for visual simplicity. Paste into the Password window in the Control Solutions device. Then click Update Server, and to make the changes persistent, follow that by going to the File Manager page and saving your configuration file.



Email Server Setup: Update Server

SMTP Host: smtp.gmail.com

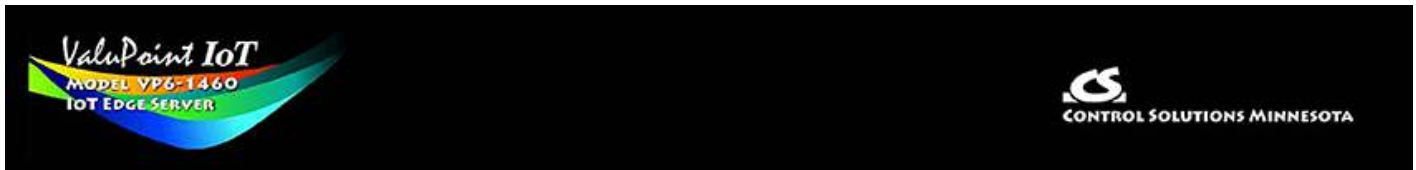
SMTP Port: 465

User Name/Email: @gmail.com

Password: ●●●●●●●●●●●●●●●

"From" Name: MQ-61A Event Test

Note that the "From" name is arbitrary. You should now be able to send email using Gmail.



11. Configuring Local Data Logging

There are two ways of going about data logging with this IoT Device. One is via the cloud (explained in other sections). The other is to log data to a local CSV file and have that file emailed to you periodically. This section pertains to local data logging and emailed CSV files.

11.1 Selection of Data Points

Selection of data points is pretty easy. Every register you have created shows up on the list here. Simply check off those registers you want to log. Go to the File Manager page and save your configuration after making these selections.

Local Register#	Header Label/Register Name	Include in CSV File
27	Data Value 1	<input checked="" type="checkbox"/>
28	Data Value 2	<input checked="" type="checkbox"/>
29	Data Value 3	<input type="checkbox"/>
30	Data Value 4	<input type="checkbox"/>
31	Data Value 5	<input checked="" type="checkbox"/>
32	Data Value 6	<input checked="" type="checkbox"/>
33	Data Value 7	<input type="checkbox"/>
34	Data Value 8	<input type="checkbox"/>
35	Data Value 9	<input type="checkbox"/>
36	Data Value 10	<input type="checkbox"/>

Note the Logging Enabled check box in the upper left corner. You must disable logging while making changes to logging parameters or register selections. Then check the Logging Enabled box and click Update to enable logging. The Logging Enabled state is retained through power outages, and logging will resume when power is restored if it was enabled to begin with.

11.2 Log Rate and File Send

Once you have selected which registers to record, this is where you decide how often to record them, and when to send the log file to yourself via email. This section of the screen appears right below the list of registers above.

Log Frequency:

Log every 10 minutes

Log every 10 minutes while event 0 is true, otherwise log every 0 minutes

Commit every 6 hours and upon event transition to false. Anticipated file size (KB) 25 (25KB min, 500KB max)

Email file to user group (1..5) 1 using template 1 at 12:00

Daily

Weekly on day 5 (1=Sunday, 2=Monday, ... 7=Saturday)

Upon event transition to false

Data logger status: 0 File system error code: 0 Records logged: 0
Time since last email (HH:MM:SS) (no email) Email error code: 0

Purge log files: Confirm by entering root password: [masked] Delete All

Select the first "Log every" line to always log at strictly the same rate.

Select the second "Log every" line and complete the rest of the line if you wish to log at one rate most of the time, but log at a different (usually faster) rate while some event of interest is taking place. A typical example of this is that you don't really need to record oil pressure very often for an engine that isn't running, but when it is running, you want to see data much more often. So you would create an event (that doesn't necessarily email any notification) that simply tells you when that engine is running based on reading a register somewhere.

Logging will normally take place every N minutes as configured. However, if the log rate is exactly 60 minutes, then the logging is synchronized with real time, and each log record will be recorded every hour on the hour.

Log and Commit are two different things. As data is logged, it is stored in a temporary file in volatile memory. Then, periodically, it will be committed to the Flash file system. The purpose for doing this is that the Flash memory has a finite lifetime measured in write cycles. You do not want to abuse the write cycles if you want years of life out of this device. The Commit will take place periodically every few hours as configured. In addition, if the log rate had been altered as the result of an event, then when that event is over with, another Commit will be done. If you are highly concerned about losing data not committed, then it is recommended that you power this IoT device from a UPS.

There are two ways to receive your log files. The easy way is to just have them automatically emailed to you. If that isn't an option, then you can log into the web UI, go to the File Manager page, and retrieve them there. Set the file filter to *.csv and click Filter. Find the file of interest in the drop-down list, and then click View. In most cases, your browser will offer you the option of saving the file or opening it in your spreadsheet program. You could also use FTP to retrieve your files.

As the Flash file system fills up, the system will automatically delete the oldest files and it can only assume that either they were emailed to you or you logged in and retrieved them.

The anticipated file size is an estimate of the size of the file you think might be emailed each time. See additional comments below.

Your log file will be emailed to the user group given, and using the template number given. This will be done at the time given in 24-hour format. A Commit will be made automatically before sending the file.

You may elect to have the log file emailed daily, weekly, or upon event transition to false. If weekly, select which day of the week you want the log to be sent. The "upon transition" may be used at the same time as daily or weekly. If you select both daily and weekly, it will automatically be just daily. The event transition refers to the event noted above that causes a different log rate to be in effect. When used at the same time as daily or weekly, the "upon transition" means "in addition to" daily or weekly.

The time since the most recent emailing of a log file is noted. If there is a non-zero error code, it will pertain specifically to the data log email, and those are explained on the Recipients page.

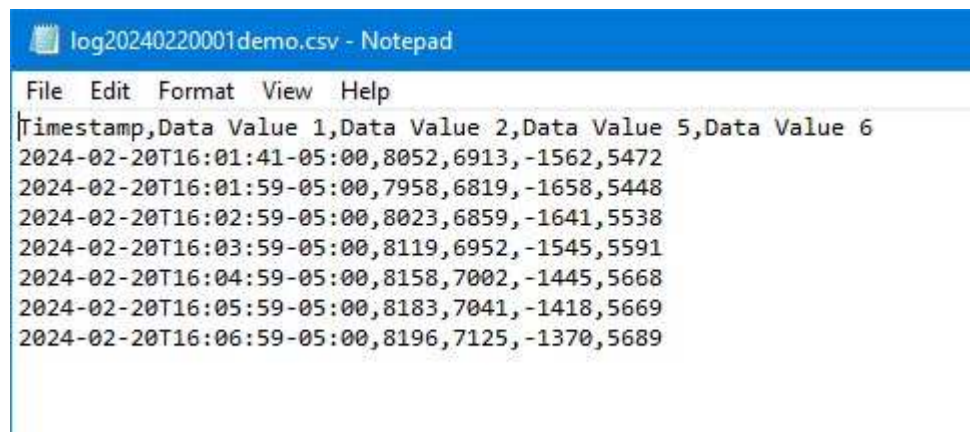
If you have reason to delete all old log files, enter the root password and click Delete All.

11.3 CSV File Format

There will be one column in the file for each register selected on the register list. No column will be allocated for non-selected registers. The first column is always timestamp and is included automatically. The first line in the file will be a header line made up of the register names of each of the logged registers. These are the names displayed on the Local Registers page. Following the header line, one line of data will be recorded every so often as configured above. Data values are separated by commas (hence the CSV notation for Comma Separated Values).

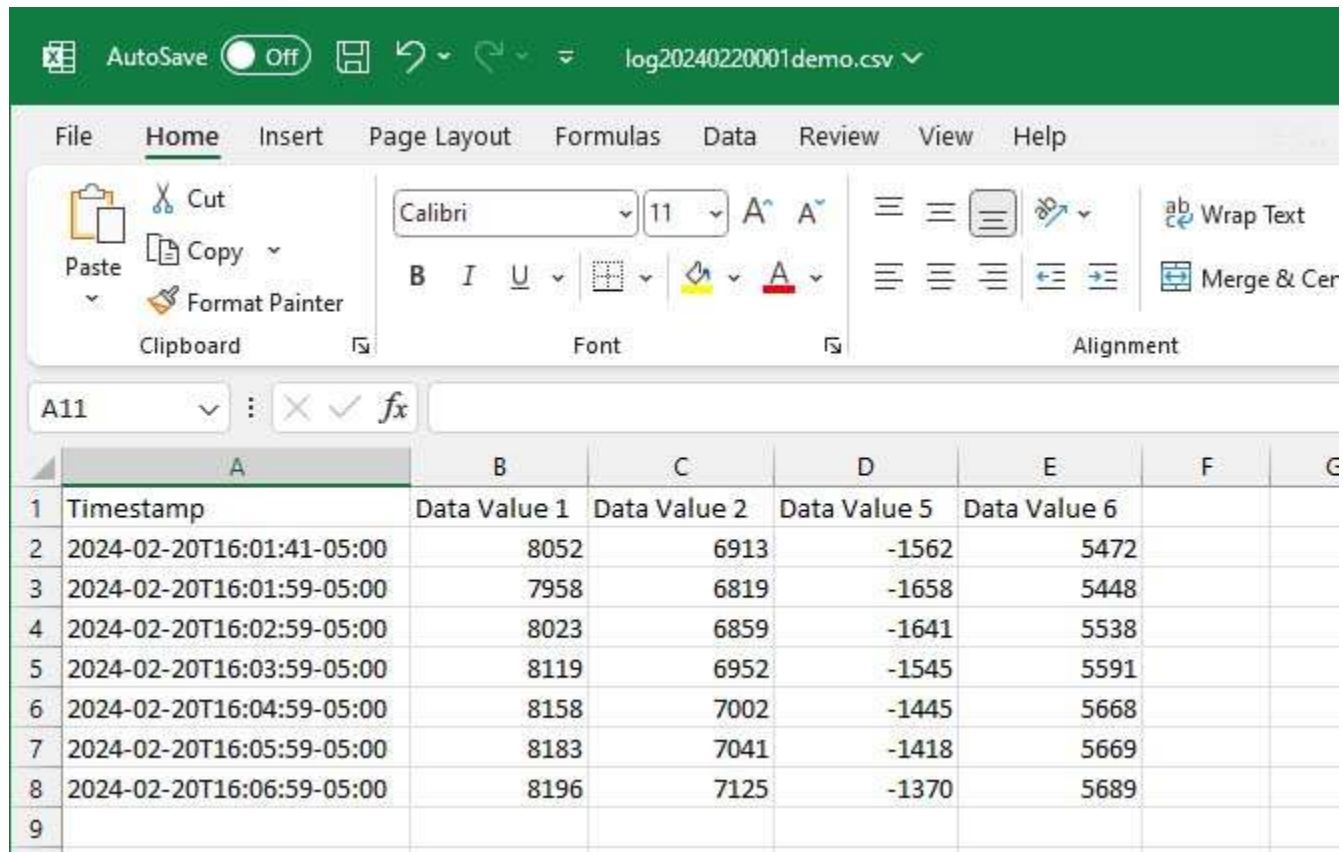
The header line is recorded one time when a new log file is created. Therefore, if you were previously logging data and then change the register selections and promptly resume logging, the data logged now will not correspond to the originally logged header file. To avoid this, retrieve your old log files, then delete all the old log files to force a new file to be created. Normally, if logging is interrupted, logging will resume writing to the same log file previously in use, and this includes when interrupted to make configuration changes.

A snippet of a sample file is illustrated below in raw text form.



```
log20240220001demo.csv - Notepad
File Edit Format View Help
Timestamp,Data Value 1,Data Value 2,Data Value 5,Data Value 6
2024-02-20T16:01:41-05:00,8052,6913,-1562,5472
2024-02-20T16:01:59-05:00,7958,6819,-1658,5448
2024-02-20T16:02:59-05:00,8023,6859,-1641,5538
2024-02-20T16:03:59-05:00,8119,6952,-1545,5591
2024-02-20T16:04:59-05:00,8158,7002,-1445,5668
2024-02-20T16:05:59-05:00,8183,7041,-1418,5669
2024-02-20T16:06:59-05:00,8196,7125,-1370,5689
```

When opened with a spread sheet program, it may look like this:



The screenshot shows a spreadsheet application window titled 'log20240220001demo.csv'. The ribbon includes 'File', 'Home', 'Insert', 'Page Layout', 'Formulas', 'Data', 'Review', 'View', and 'Help'. The 'Home' ribbon is active, showing options for Clipboard (Paste, Cut, Copy, Format Painter), Font (Calibri, size 11, bold, italic, underline, color), and Alignment (Wrap Text, Merge & Center). The active cell is A11. The data table below is as follows:

	A	B	C	D	E	F	G
1	Timestamp	Data Value 1	Data Value 2	Data Value 5	Data Value 6		
2	2024-02-20T16:01:41-05:00	8052	6913	-1562	5472		
3	2024-02-20T16:01:59-05:00	7958	6819	-1658	5448		
4	2024-02-20T16:02:59-05:00	8023	6859	-1641	5538		
5	2024-02-20T16:03:59-05:00	8119	6952	-1545	5591		
6	2024-02-20T16:04:59-05:00	8158	7002	-1445	5668		
7	2024-02-20T16:05:59-05:00	8183	7041	-1418	5669		
8	2024-02-20T16:06:59-05:00	8196	7125	-1370	5689		
9							

11.4 Anticipated File Size

The anticipated file size is initially just an estimate of the size of the file you think might be emailed each time. Once you start to see what the normal file size is, set the anticipated file size to something just beyond that size. What this does is cause the system to delete enough old files to make this amount of free space for the new log file about to be started. This action will take place each time the log file is sent and the system begins creating the next new log file.

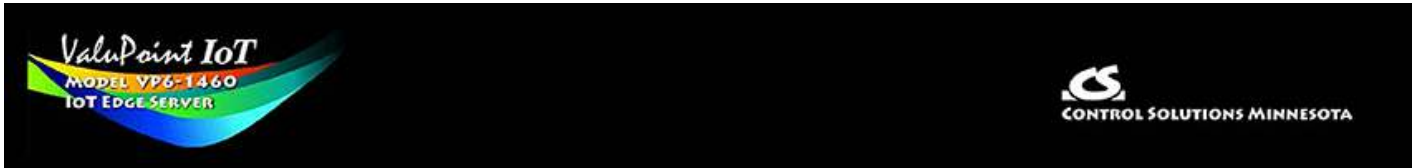
If the anticipated file size was too small, and the system runs out of room, it will commit and send the file even if it was not normally time to do so. It will then repeat the process of trying to delete some old files and resume logging. If your log files are showing up more often than the schedule you anticipated, it is likely due to running out of room.

Most often, running out of room will happen in the middle of an attempt to commit. If this happens, the file as already committed will be sent, then old files will be deleted to make room, then the commit will be repeated so that no data is lost. This can mean that some data points will be recorded in both the file just sent and the new file just created.

If Internet service is unavailable when the ValuPoint attempts to send a log file, it will continue to retry. If Internet service has not been restored by the time the next log file should be sent, then when Internet finally is restored, only the most recent log file will be emailed. You will need to log into the web UI to manually retrieve log files if files

were skipped due to extended Internet outage.

As for calculating your initial estimate, there is no precise formula for doing that. Simply multiplying point count by some fixed number will be inaccurate because different data types end up formatted differently. Simply try to estimate the number of characters per line, multiplied by how many lines there should be at the given log rate by the time the file is sent.



12. Configuring the Scheduler

The ValuPoint becomes more useful when control functions can be combined with monitoring. One element of control that is often useful is the ability to schedule things to happen at certain times on certain days. The scheduler makes that possible.

Scheduling is done in a very generic and simple way. A register you select will change value according to a schedule you provide. From there, you can use the client to write that register to some external Modbus device to cause action according to your schedule.

The scheduler does require access to an SNTP server in order to know what the current time and date are. Be sure to configure NTP on the Network setup page. If SNMP is not an option, you can also use the internal battery backed real time clock but it will be up to you to be sure it is set correctly.

12.1 Weekly Schedule

The weekly schedule allows you to specify that something should happen at a certain time of certain days of the week. It can be one day, multiple days, or every day.

The days of the week start with Sunday in the left column. Simply check the boxes for those days you want action. Then select on and off time of day using 24-hour format. Select a register number, and its "on" and "off" value.

The "on" state will be that period that falls between On Time and Off Time. Any other time is "off". If multiple lines are used for the same day and same register, they should be organized with later times last and they will be processed sequentially.

Using the example illustrated below, local register 27 will be set to a value of 10 from 10:00AM until noon on Sunday, and be set to a value of 2 at all other times. And so forth.

Valupoint IoT
MODEL VP6-1460
IOT EDGE SERVER

CONTROL SOLUTIONS MINNESOTA

Local Data Modbus IoT Cloud System

System Setup Programming Scheduler

Weekly Schedule On Demand Holidays

Showing 1 to 11 of 11 Update < Prev Next >

#	S...M...T...W...T...F...S	Holidays	On Time	Off Time	Register Number	"On" Value	"Off" Value	Register Name
1	<input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	Holidays	10:00:00	12:00:00	27	10.00000	2.000000	Data Value 1
2	<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	Holidays	10:00:00	12:00:00	28	10.00000	2.000000	Data Value 2
3	<input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	Holidays	10:00:00	12:00:00	29	10.00000	2.000000	Data Value 3
4	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	Holidays	10:00:00	12:00:00	30	10.00000	2.000000	Data Value 4
5	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	Holidays	10:00:00	12:00:00	31	10.00000	2.000000	Data Value 5
6	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	Holidays	10:00:00	12:00:00	32	10.00000	2.000000	Data Value 6
7	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/>	Holidays	10:00:00	12:00:00	33	10.00000	2.000000	Data Value 7
8	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	Holidays	10:00:00	12:00:00	34	10.00000	2.000000	Data Value 8
9	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>	Holidays	10:00:00	12:00:00	35	10.00000	2.000000	Data Value 9
10	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>	Holidays	DUSK-30	DAWN+30	36	10.00000	2.000000	Data Value 10
11	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	Holidays	0:00:00	0:00:00	0	0.00	0.00	

Events Enabled: 11 ReSync Insert Delete

Note the ReSync button at the bottom. If you have made changes to the scheduler, ReSync will cause everything about the schedule to be re-evaluated and registers updated accordingly. Normally, registers are written only when the schedule says it is time to change. The evaluation is made at the start of the configured time period. Therefore, if you have made a new schedule entry that says a register should be "on" now, you will need to hit the ReSync button to cause that to happen now.

Click Update to register your changes. The "Events Enabled" simply sets the scope of the web page display. If you are just starting out and want to see a page of 10 unused entries, set this to 10 and update. If you have many entries, use Next and Prev to scroll through the list. Insert will insert a new blank entry before the entry number in the Showing box at the top. Delete will delete the entry number you enter in the Showing box at the top.

12.2 On Demand Scheduled Events

The scheduler also provides the opportunity to schedule something to happen just one time on a given day or days. Instead of day of week, a date is provided here. Other than selection of day, the On Demand scheduler works the same as Weekly scheduler (except there are no holidays for On Demand).

Using the example illustrated below, local register 27 will be set to a value of 100 starting at 3:00PM February 27, 2024, and remain at that value until 10:00AM February 28. At all other times, local register 27 will be set to 1. And so forth.

Weekly Schedule		On Demand		Holidays					
#	On Time	On Date Y-M-D	Off Time	Off Date Y-M-D	Register Number	"On" Value	"Off" Value	Register Name	
1	15:00:00	2024-02-27	10:00:00	2024-02-28	27	100.0000	1.000000	Data Value 1	
2	10:00:00	2024-02-27	15:00:00	2024-02-28	28	200.0000	2.000000	Data Value 2	
3	14:00:00	2024-02-27	14:30:00	2024-02-27	29	300.0000	3.000000	Data Value 3	
4	15:00:00	2024-02-27	15:30:00	2024-02-27	29	310.0000	4.000000	Data Value 3	
5	16:00:00	2024-02-27	16:30:00	2024-02-27	29	320.0000	5.000000	Data Value 3	
6	0:00:00	0000-00-00	0:00:00	0000-00-00	0	0.00	0.00		

Showing 1 to 6 of 6

Update < Prev Next >

Commands Enabled: 6

Insert Delete

Entries applied to the same register number will be processed sequentially. Register 29 in the above example will be set to a value of 3 prior to 2:00PM Feb. 27. Then from 2:00PM to 2:30PM on Feb. 27, it will be set to a value of 300. From 2:30PM to 3:00PM, the value will be 3. From 3:00PM to 3:30PM, the value will be 310. From 3:30PM to 4:00PM, the value will be 4. From 4:00PM to 4:30PM, the value will be 320. Any time after 4:30PM Feb. 27, the value will be 5. On October 1, the value in register 29 will still be 5.

Click Update to register your changes. The "# Commands Enabled" simply sets the scope of the web page display. If you are just starting out and want to see a page of 10 unused entries, set this to 10 and update. If you have many entries, use Next and Prev to scroll through the list. Insert will insert a new blank entry before the entry number in the Showing box at the top. Delete will delete the entry number you enter in the Showing box at the top.

12.3 Holidays

Sometimes you want a weekly schedule to not apply on a holiday, or maybe you want something to only happen on a holiday (although that would be nearly the same as On Demand). The holiday processing in the scheduler allows exceptions to the weekly schedule.

Start by creating a Holiday on the Holidays tab. Give it a name, start time and date, and end time and date. Most often the start time for a holiday will be 0:00:00 and end time will be 23:59:59 so that it means "all day". You may create up to 32 holidays.

Weekly Schedule		On Demand		Holidays	
<input type="button" value="Update"/>					
#	Holiday Name	On Time	On Date Y-M-D	Off Time	Off Date Y-M-D
1	Test Holiday	0:00:00	2024-02-14	23:59:00	2024-02-14
2		0:00:00	0000-00-00	0:00:00	0000-00-00
3		0:00:00	0000-00-00	0:00:00	0000-00-00

To incorporate a holiday into a weekly schedule entry, click on that line's Holidays link.

Weekly Schedule		On Demand		Holidays				
Showing 1 to 11 of 11 <input type="button" value="Update"/> <input type="button" value=" < Prev"/> <input type="button" value=" Next >"/>								
#	S...M...T...W...T...F...S	Holidays	On Time	Off Time	Register Number	"On" Value	"Off" Value	Register Name
1	<input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	Holidays	10:00:00	12:00:00	27	10.00000	2.000000	Data Value 1
2	<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	Holidays	10:00:00	12:00:00	28	10.00000	2.000000	Data Value 2
3	<input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	Holidays	10:00:00	12:00:00	29	10.00000	2.000000	Data Value 3
4	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	Holidays	10:00:00	12:00:00	30	10.00000	2.000000	Data Value 4
5	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	Holidays	10:00:00	12:00:00	31	10.00000	2.000000	Data Value 5
6	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	Holidays	10:00:00	12:00:00	32	10.00000	2.000000	Data Value 6
7	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/>	Holidays	10:00:00	12:00:00	33	10.00000	2.000000	Data Value 7
8	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	Holidays	10:00:00	12:00:00	34	10.00000	2.000000	Data Value 8
9	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>	Holidays	10:00:00	12:00:00	35	10.00000	2.000000	Data Value 9
10	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>	Holidays	DUSK-30	DAWN+30	36	10.00000	2.000000	Data Value 10
11	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	Holidays	0:00:00	0:00:00	0	0.00	0.00	
Events Enabled: 11 <input type="button" value="ReSync"/>						<input type="button" value="Insert"/> <input type="button" value="Delete"/>		

The available holidays will be listed. To add a holiday, click on the holiday in the Available list and click the Add button. To remove a holiday previously added, click on the holiday in the Selected list and then click the Remove button. Once you have added a holiday or two, select whether to include or exclude.

The effect of exclude is to temporarily, effectively, uncheck that day of the week. The effect of include is to temporarily, effectively, check that day of the week. In the example below, regardless of what day of the week it is, if this day happens to be the holiday, the "On" value will not be applied between 10:00AM and noon.

Weekly Schedule		On Demand	Holidays				
		Showing 9 of 11		Update		< Prev Next >	
#	S...M...T...W...T...F...S	On Time	Off Time	Register Number	"On" Value	"Off" Value	Register Name
9	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>	10:00:00	12:00:00	35	10.00000	2.000000	Data Value 9
Available Holidays		Selected Holidays				<input checked="" type="radio"/> Exclude <input type="radio"/> Include	
Test Holiday		Test Holiday		Add>			
				<Remove			

Note that in the following example, no days of the week are selected but a holiday is selected as included. This is effectively an On Demand scheduled event for that holiday. The "On" value will be applied on this holiday, regardless of day of week, between 10:00AM and noon (assuming the holiday is defined as all day - if the holiday starts at 3:00PM, then the "On" value would not be applied and this entry in the schedule will never do anything.)

Weekly Schedule		On Demand	Holidays				
		Showing 8 of 11		Update		< Prev Next >	
#	S...M...T...W...T...F...S	On Time	Off Time	Register Number	"On" Value	"Off" Value	Register Name
8	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	10:00:00	12:00:00	34	10.00000	2.000000	Data Value 8
Available Holidays		Selected Holidays				<input type="radio"/> Exclude <input checked="" type="radio"/> Include	
Test Holiday		Test Holiday		Add>			
				<Remove			

12.4 Astronomical Clock

If you were looking closely at the first example in this section, you may have noticed one peculiar entry.

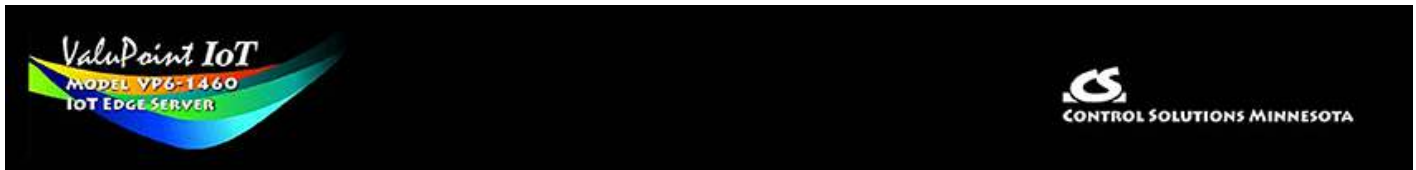
9	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>	Holidays	10:00:00	12:00:00	35	10.00000	2.000000	Data Value 9
10	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>	Holidays	DUSK-30	DAWN+30	36	10.00000	2.000000	Data Value 10
11	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	Holidays	0:00:00	0:00:00	0	0.00	0.00	

Events Enabled: 11 ReSync Insert Delete

Suppose you are scheduling lights to come on when it gets dark outside. One way of doing that is with a light sensor. Another way is by scheduling, but then you have to keep changing the on and off times throughout the seasons. The astronomical clock feature of this scheduler will keep changing the on and off times for you when you use "DUSK" and "DAWN" as entries. In the example above, register 36 will be set to a value of 10 thirty minutes before sundown, and returned to a value of 2 thirty minutes after sunrise. The more likely scenario would be an on value of 1 and off value of 0 to

switch a switch somewhere.

Note that in order for the astronomical clock to work correctly in your location, you must set the latitude and longitude for the location on the Network setup page, NTP section. You will also see the currently calculated sunrise and sunset times displayed there.



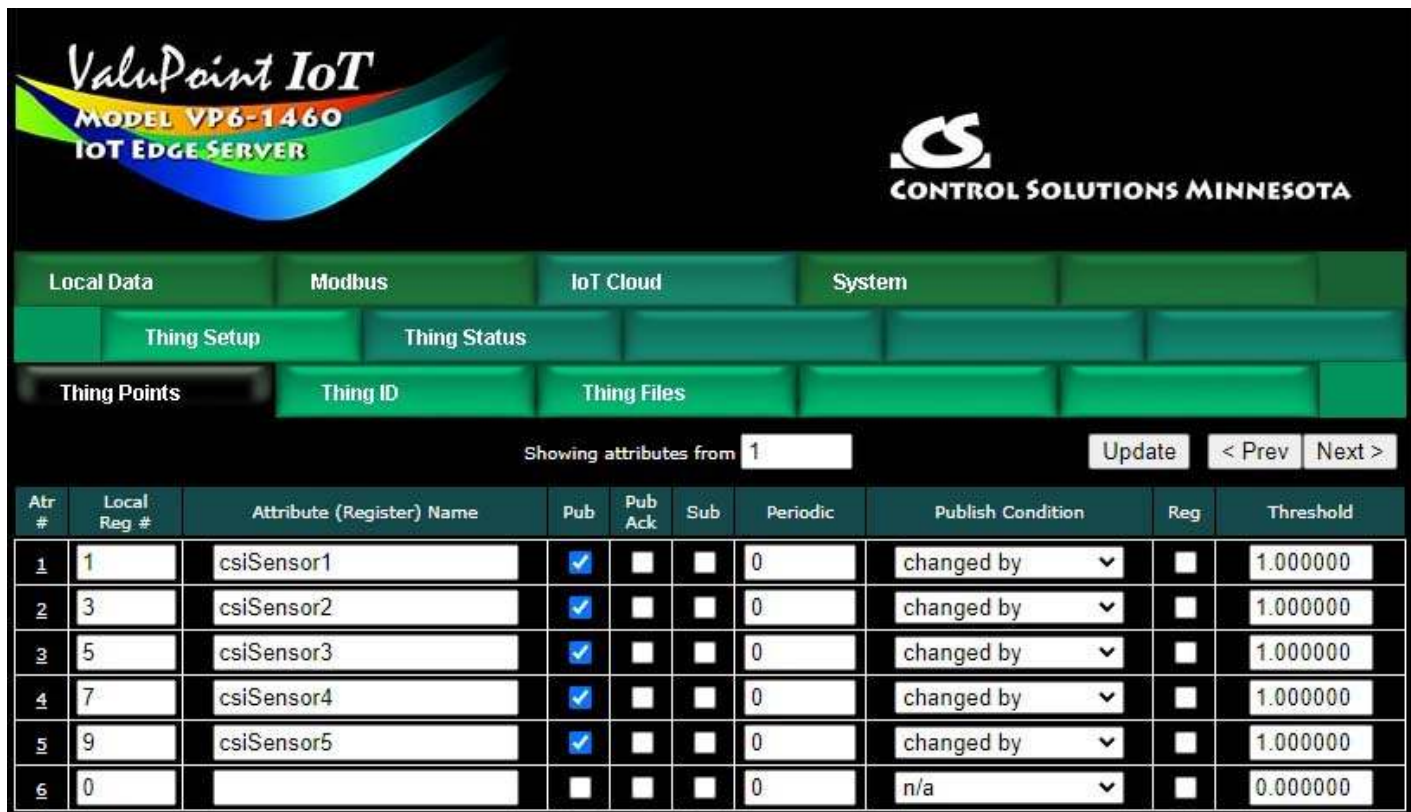
13. Configuring the IoT Client

The ValuPoint IoT Edge Server functions as an MQTT client for purposes of transmitting data to MQTT brokers such as AWS IoT, Mosquitto MQTT, or Thingsboard. The pages used to configure the ValuPoint IoT Client's connection to Amazon Web Services (AWS) are detailed in this section. This section provides a reference for elements of the pages, but to gain an understanding of the overall flow of configuring both the IoT Client and the AWS IoT features, you will want to refer to Sections 14 and 15 of this user guide.

13.1 Thing Points or Attributes

The ValuPoint is used to turn physical I/O as well as any Modbus device into "things" for the Internet of Things. The AWS server only knows that we have a "thing" of some sort, and the MQTT protocol used by the Internet of Things knows how to exchange messages containing attributes which have values. The ValuPoint maps Modbus registers (which may be connected to local I/O) to attributes, and the Modbus register content is treated as the attribute's value. The mapping also includes mapping Modbus register numbers to attribute names. Each Modbus register becomes a data point that is treated as an attribute by MQTT.

The Thing Points page shows a list of all current attributes, i.e., local registers that have been mapped as an attribute for our "thing". You can create very simple publish and subscribe rules on this tabular list of attributes, but for maximum flexibility, you will want to review each attribute individually.



Valupoint IoT
MODEL VP6-1460
IOT EDGE SERVER

CONTROL SOLUTIONS MINNESOTA

Local Data Modbus IoT Cloud System

Thing Setup Thing Status

Thing Points Thing ID Thing Files

Showing attributes from 1 Update < Prev Next >

Atr #	Local Reg #	Attribute (Register) Name	Pub	Pub Ack	Sub	Periodic	Publish Condition	Reg	Threshold
1	1	csiSensor1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by	<input type="checkbox"/>	1.000000
2	3	csiSensor2	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by	<input type="checkbox"/>	1.000000
3	5	csiSensor3	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by	<input type="checkbox"/>	1.000000
4	7	csiSensor4	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by	<input type="checkbox"/>	1.000000
5	9	csiSensor5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by	<input type="checkbox"/>	1.000000
6	0		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	n/a	<input type="checkbox"/>	0.000000

Click on the attribute number in the first column to open the expanded view of the attribute and its publish and subscribe rule. The attribute number has no significance other than the order in which it will be listed in the message when multiple attributes are included in the same message.

Begin by selecting the local register this attribute rule will be associated with. This will be one of the register numbers listed on the Local Registers page when you created registers in Section 4 of this user guide. The rules for naming MQTT attributes are stricter than the rules for naming local registers. Therefore, the name will be "cleaned up" as needed when you select the register. Attribute names can have no embedded spaces and no special characters - only letters and digits. The names should also be unique to avoid confusion.

If you enter only a number, the name already provided on the Local Registers page will automatically be used. If you enter a name here and it differs from the name previously assigned on the Local Registers page, the name will be changed both here and on the Local Registers page.

Select Publish if you wish to publish this point. Publish means send data from the

ValuPoint to the AWS server, and is the type of action you would associate with a sensor.

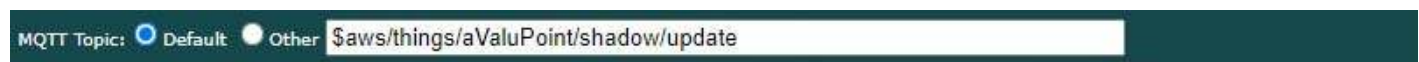
Select whether or not you wish to publish with acknowledgement required. This is referred to in MQTT terms as Quality of Service (QOS). If "Ack required" is selected, then the ValuPoint will repeatedly retry publishing until the server responds with an acknowledgement.

You will normally publish data points as "Reported". If, however, you are publishing to the shadow object of another ValuPoint or similar IoT device, and your intent is to set a register value in that remote device via the AWS server, then you would publish as "Desired".



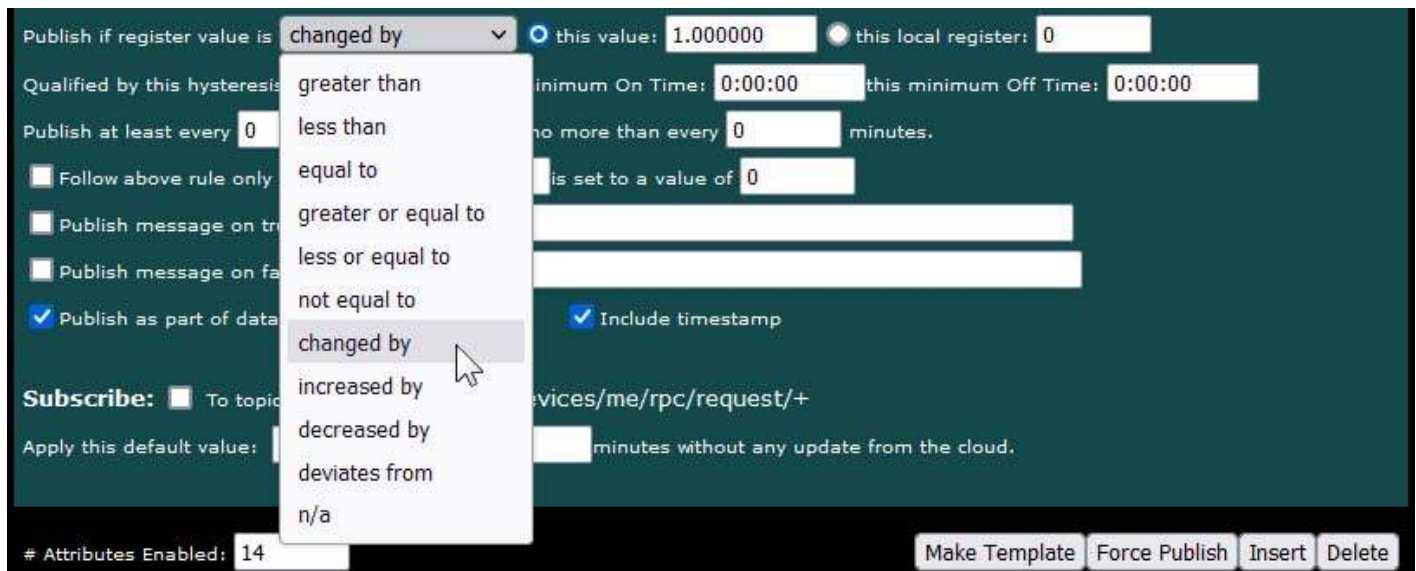
The MQTT topic displayed here is used as the topic when the Publish is invoked, and the attribute name and value are considered the "payload" published to this topic. The default topic will publish to the shadow object for this ValuPoint that you will set up on the AWS server (see Section 14).

If you wish to publish to a topic other than the default topic, enter that topic here and select "Other".



To publish periodically and only periodically, skip the threshold test and value. Enter a number of minutes for "Publish at least every" and you're done.

To publish upon a given condition, select a test from the drop-down list. The test will be applied to the threshold value given as "this value", or to the threshold value currently found in the local register given if that option is selected instead. Normally, the attribute will be published once when the test first transitions to "true", and published again when the test transitions back to "false". If AWS is being used to notify users about an alarm condition, then the publish rule might be "greater than" some threshold.



Some tests need further explanation. The "Changed by" test means amount of change since the last publish of this attribute. If the local register has changed by the value specified as "this value" or the value contained in the local register referenced, the test is true. The "Changed by" value can be an increase or decrease. To publish only upon increase or decrease since the last publish, select those tests instead. The "Deviates by" uses a special application of the hysteresis value. If the present value of the local register deviates from the threshold by the margin set as hysteresis, then this test will be deemed to be "true". This amounts to a combined greater than and less than in the same test.

Tests such as "changed by" will publish each time the attribute changes by that value. There is no static state "true" or "false" for a "changed by" rule. In addition, the "changed by" rule can have a threshold value of zero - this will cause the attribute to be published any time the local register is updated regardless of its new value (and regardless of whether changed). Exercise caution when using "changed by zero" for a register that is being read every few seconds by Modbus - be sure you really want to send data to AWS this often.

Qualifications are optional, and enabled only when values are nonzero. How hysteresis is applied depends on the comparison. For a test that becomes true if greater than, the test will not return to false until the local register is less than the test value by a margin of at least this hysteresis value. If a test becomes true if less than, it will not return to false until the local register is greater than the test value by a margin of at least this hysteresis value.

On time and off time, if specified, determine how long the condition must be true (on time) or false (off time) before the true or false response is actually taken. Times are given in HH:MM:SS format (hours, minutes, seconds). On/off time qualifications should not be used with "changed by" or other transition type tests. These time qualifications can only apply to static tests such as "greater than".



Publish at least every N minutes will result in periodic publishing regardless of any

conditional testing. Periodic publishing is disabled by entering zero here.

Publish no more than every N minutes will limit the number of times the Thing is permitted to publish. Regardless of what condition exists, it will not be published until this amount of time has expired since the last publish. This throttle effect is disabled by entering zero here. If this attribute is included in a data set being published as result of another rule being fully satisfied, this attribute will be included regardless of time since last publish.

Publish at least every minutes. Publish no more than every minutes.

You have the option of enabling publishing of this attribute only when a selected local register contains a given value. Any local register may be used as the enable register. The optional enable register applies to publishing based on this rule. If this attribute is included in a data set that is successfully triggered by another rule, then this attribute will be included regardless of enable register value

Follow above rule only if local register is set to a value of

You have the option of publishing fixed messages instead of the register value. Check the applicable "Publish message" boxes and provide a message. The expression "%s" will be replaced by an ASCII representation of the present value of the local register if %s is found in the string.

Publish message on true:
 Publish message on false:

Data sets in the AWS cloud should be viewed like a spread sheet. For best practical use, you want to populate all columns each time you add a new row. In order to make this happen, you need to include all related attributes in the same publish message sent to the server. To cause the attributes of interest to become associated, check "Publish as part of data set" and enter a number. Upon any attribute rule triggering a publish, all attributes with the same data set number will be included in the message.

To include the timestamp as a data element, check "with timestamp".

Publish as part of dataset number: Include timestamp

To allow this attribute to take on new values from other sources by subscribing to other resources in the cloud via this Thing's shadow, check Subscribe.

The subscribe topic will be the topic selected for this attribute; however, only values published as "desired" for this attribute will be acted upon. Any value published to this attribute's topic as "desired" will result in changing the value in this local register. Enter a topic index to select the topic - enter the subscribe topics on the Thing ID page.

You have the option of setting the local register to a default value if no subscription value is received within some number of minutes. Enter both the default value and

timeout. If the timeout is zero, the default setting is disabled.

WARNING: Be very cautious about selecting both publish and subscribe at the same time. It is possible to configure an endless loop of continuously publishing to yourself. There are multiple reasons you don't want this to happen.



Subscribe: To topic index: \$aws/things/myFirstThing/shadow/update
Apply this default value: after minutes without any update from the cloud.

Click the Update button to register any changes you have made. The Update button moves data from your browser to the ValuPoint. **IMPORTANT:** To make the changes effectively permanent, you also need to go to the File Manager page and save your configuration as an XML file.

The Prev/Next buttons simply scroll through the list of attribute rules.



Thing Points **Thing ID** **Thing Files**
Attribute #
Associate local register # named with this IoT attribute.
Attributes Enabled:

Insert will insert a new attribute before the attribute number shown, and is used for placing attributes between existing attributes. It is not necessary to use Insert to add attributes to the bottom of the list or to redefine any attribute presently having zero for an "associate" register. Attribute numbers work like row numbers in a spreadsheet. If you insert an attribute, existing attributes slide down the sheet and get a new number. Likewise if you delete an attribute, the rest of the attributes slide up the sheet and get new attribute numbers.

Delete will remove the attribute number shown in the "Showing" box. Entering zero as the "Associate" register will also effectively delete the attribute even though it will still appear in the list until deleted. Unused attributes at the end of the list will always show zero as the associate register. If you wish to prevent these from being displayed, reduce the number of attributes enabled.

The number of attributes enabled simply limits the scope of attribute review so that you do not have to review a lot of unused attributes.

Click Make Template to fill the "Last Pub" buffer on the Test page with a template of what would be published if this point were published. You may then copy/paste this into a file as needed to upload a JSON example to the AWS server when setting up SNS notifications. Click Make Template here, then go to the test page and click Last Pub.

Click Force Publish to do a one-time publish of the attributed shown on this page, and do so without regard to the rules or conditions. This will force an immediate publish of

this attribute (if Publish is enabled). You can then check the result on the Thing Status :: Test page.

The Update and Prev/Next buttons on the Thing Points tabular summary page have the same effect as noted above.

Thing Points									
Thing ID									
Thing Files									
Showing attributes from 1									
Update < Prev Next >									
Atr #	Local Reg #	Attribute (Register) Name	Pub	Pub Ack	Sub	Periodic	Publish Condition	Reg	Threshold
1	1	csiSensor1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by ▼	<input type="checkbox"/>	1.000000
2	3	csiSensor2	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by ▼	<input type="checkbox"/>	1.000000
3	5	csiSensor3	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by ▼	<input type="checkbox"/>	1.000000
4	7	csiSensor4	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by ▼	<input type="checkbox"/>	1.000000
5	9	csiSensor5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by ▼	<input type="checkbox"/>	1.000000
6	0		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	n/a ▼	<input type="checkbox"/>	0.000000

Referring to the preceding detailed descriptions, select a local register number to be published or subscribed, and allow the ValuPoint to retrieve the name for you or enter a new name following the guidelines noted above.

Select "Pub" for publish, "Pub Ack" for publish with acknowledge required, or "Sub" to subscribe. To set the "Publish at least ever N minutes", enter the number of minutes in the Periodic column.

Select a conditional test, and provide a threshold. If the threshold should be retrieved from another local register, check the "Reg" box and provide a local register number instead of fixed value in the Threshold column.

13.2 Thing ID and Subscribe Topics

The items on the top half of the Thing ID page are important elements of establishing your ValuPoint's connection to the AWS server. The Subscribe Topics are used if subscribing, and are not used to publish.

ValuPoint IoT
MODEL VP6-1460
IOT EDGE SERVER

CONTROL SOLUTIONS MINNESOTA

Local Data Modbus IoT Cloud System

Thing Setup Thing Status

Thing Points Thing ID Thing Files

Update

Server Host Name

Server Port Disable SSL Disable SSL certificate verify

Thing Name / Client ID

Username

Password

Features Enabled: AWS IoT Core Complex JSON Thingsboard RPC

IoT Engine Status Enabled (See IMPORTANT Note Below)

Subscribe Topics:

Topic 0

Topic 1

Topic 2

Topic 3

Topic 4

The Server Host Name will be provided to you by Amazon. Refer to Section 14 to see where you find this. The default port normally used for secure MQTT is 8883. Use this port unless instructed otherwise by Amazon.

The Thing Name is a name you will assign. The only important guideline here is make sure the Thing Name you provide on this page in the ValuPoint is the same name you provided to Amazon when setting up your Thing on the AWS server. The thing name should be unique, but AWS will enforce that for you along with any other naming restrictions. Create the thing on the AWS server first, then enter the thing name here.

The interaction between the ValuPoint and the AWS server will not begin until you set the IoT Engine Status to Enabled.

IMPORTANT: BEFORE ENABLING, make sure you have established your Amazon account and made the corresponding entries above, uploaded valid SSL certificates, and set up attributes in your Thing Shadow via your Amazon account.

DO NOT ENABLE the IoT Engine with invalid configuration information. Doing so for an extended period may get you blocked by the Amazon servers. If you see errors reported by Amazon on the Thing Status page, it is a good idea to disable the IoT

Engine while you figure out why the error(s) occurred.

When connecting to AWS, select AWS IoT Core and Complex JSON. Do not select Thingsboard RPC unless connecting to Thingsboard instead of AWS.

Username and password are not used for connecting to AWS. Your SSL certificates define your identity instead of a username and password when connecting to AWS. The username and password may be used for MQTT services other than AWS.

Subscribe Topics:

Topics that Thing Points may subscribe to are defined on this page. The topics for publishing are arbitrary and you may define as many publish topics as you like on a point by point basis. However, subscribe topics require keeping track of callback handling, and therefore the subscribe topics must be declared on the above list and then referenced by index number when used to subscribe individual points. Topic 0 will default to being the shadow/update topic, and it is quite possible you will need no additional topics.

13.3 Thing Files

The connection between the ValuPoint and the AWS server is a secure connection requiring valid SSL certificates. Certificates may be optionally used with other services as well.



The screenshot shows the ValuPoint IoT Edge Server web interface. The top left features the logo for ValuPoint IoT, Model VP6-1460, IoT Edge Server. The top right features the logo for Control Solutions Minnesota. The interface has a navigation menu with tabs for Local Data, Modbus, IoT Cloud, System, Thing Setup, Thing Status, Thing Points, Thing ID, and Thing Files. The 'Thing Files' tab is selected. Below the navigation menu, there is a 'Local file directory' dropdown menu set to 'AmazonRootCA1.pem' and a 'View' button. Below this, there are three rows of input fields for certificates, each with an 'Apply' button to its left:

Apply Device Cert	f2207cfd6b098-certificate.pem
Apply Private Key	f2207cfd6b098-private.pem
Apply Root CA	AmazonRootCA1.pem

This page is where you assign the security certificates associated with your Thing. These certificates will be created for you by Amazon when you register your Thing. Download those from Amazon, upload them to this device via the File Manager page, and then select them here. Your Thing will not connect to Amazon AWS without these.

Follow directions found in Section 14 of this User Guide as well as instructions found on the Amazon AWS site regarding how to create these certificates. Be sure you have NTP set up - SSL certificates will be treated as invalid if the correct time and date are not set within the ValuPoint.

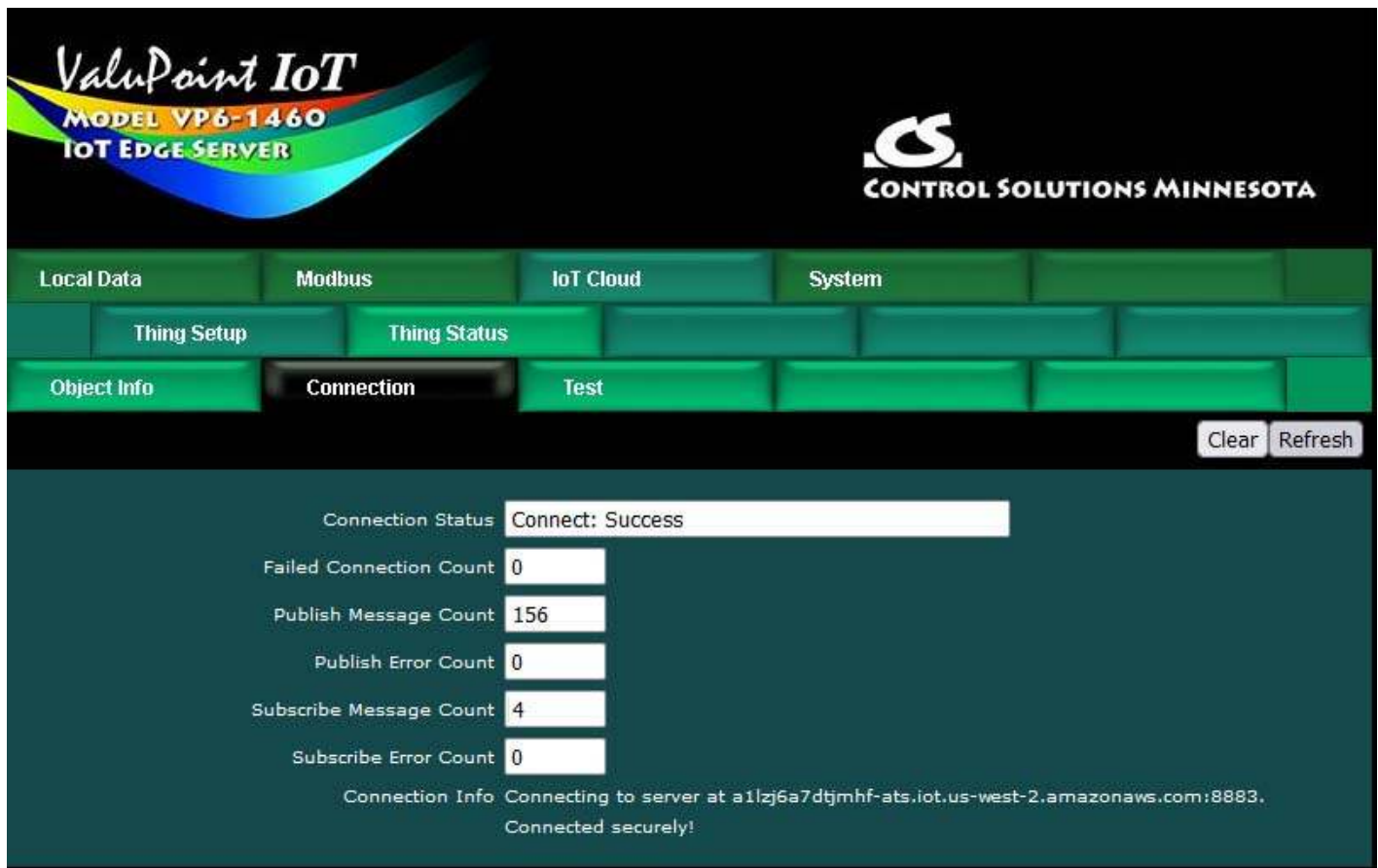
13.4 Thing Status

The most recent data exchange for each defined attribute is listed on the Object Info page. The timestamp will show "Pub@" for a published attribute, and "Sub@" when an incoming message was processed for a subscription.

The screenshot shows the ValuPoint IoT interface for Model VP6-1460. The navigation menu includes: Local Data, Modbus, IoT Cloud, System, Thing Setup, Thing Status, Object Info, Connection, and Test. The 'Object Info' page is active, showing 'Showing attributes from 1' and buttons for 'Refresh', '< Prev', and 'Next >'. The table below displays the following data:

Atr #	Attribute Name	Last Exchange Timestamp	Last Exchanged Value (or Status)
1	csiSensor1	Pub@ 2024-03-11 10:06:55	2.294626
2	csiSensor2	Pub@ 2024-03-11 10:06:55	0.005569
3	csiSensor3	Pub@ 2024-03-11 10:06:55	0.002785
4	csiSensor4	Pub@ 2024-03-11 10:06:55	0.002785
5	csiSensor5	Pub@ 2024-03-11 10:06:55	0.000000
6			

The Connection page will show the status of the ValuPoint's connection with the AWS server, along with some message statistics.



The connection status will show "Offline" when not connected because the IoT Engine is not enabled on the Thing ID page. Upon enabling the engine, there will be a delay while the ValuPoint attempts to connect. There are numerous error messages that can be potentially displayed instead of "Success". If the error indicates unable to connect, check to see that you have valid DNS server settings entered on the Network page. If you see an error such as "SSL certificate error", check to see that NTP has found the correct local time and date on the Network page. If NTP has found the correct time/date, recheck your SSL certificate setup.

13.5 Testing Thing's Connection

There are two main functions of the Test page. The most commonly used function will be to simply check the content of the most recent publish or subscribe message. The other available function is to generate arbitrary publish and subscribe message exchanges with the AWS server.

To review the most recent Publish message sent by the ValuPoint, click the Last Pub button. An example of a recent publish message is illustrated below.

The screenshot shows the 'Test' tab of the IoT Client interface. At the top, there are tabs for 'Thing Setup', 'Thing Status', 'Object Info', 'Connection', and 'Test'. Below these, there are sections for 'Subscribe:' and 'Publish:'. The 'Subscribe:' section has a text input field and 'Subscribe' and 'Unsubscribe' buttons. The 'Publish:' section has radio buttons for 'Using QOS', 'Ack not required' (selected), and 'Ack required', followed by a text input field and a 'Publish' button. A large text area displays the 'Most recent Publish or Subscribe message:' with a 'Last Pub' and 'Last Sub' button. Below this, a 'Topic:' field contains '\$aws/things/aValuPoint/shadow/update'. The message payload is shown in a code block:

```
{ "state": { "reported": { "csiSensor1": 2.294626, "csiSensor2": 0.005569, "csiSensor3": 0.002785, "csiSensor4": 0.002785, "csiSensor5": 0.000000, "LocalTime": "2024-03-11T10:06:55-05:00" } } }
```

An example of a recently received incoming message resulting from a Subscribe is illustrated below.

This screenshot shows the 'Test' tab of the IoT Client interface, focusing on the 'Most recent Publish or Subscribe message:' section. The 'Last Pub' and 'Last Sub' buttons are visible. The 'Topic:' field contains '\$aws/things/aValuPoint/shadow/update'. The message payload is shown in a code block:

```
{ "state": { "desired": { "csiActuator1": 1 } } }
```

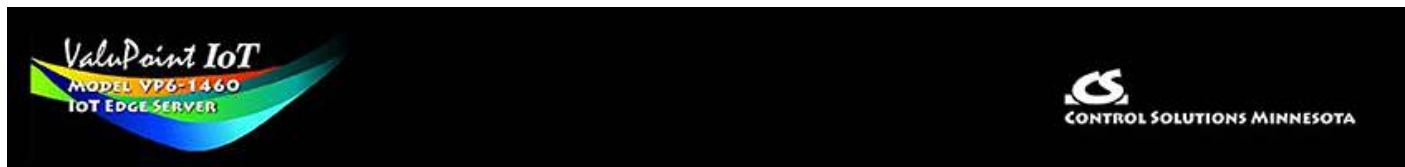
An arbitrary publish message may be sent on this page by entering a topic and payload, and then click the Publish button. This feature should only be used after you have familiarized yourself with MQTT protocol and the AWS side of this connection.

This screenshot shows the 'Test' tab of the IoT Client interface. The 'Publish:' section has radio buttons for 'Using QOS', 'Ack not required' (selected), and 'Ack required'. Below this, there is a text input field containing '\$aws/things/myFirstThing/message1' and a 'Publish' button. A large text area labeled 'Test message' is visible below the input field.

To subscribe to an arbitrary topic without configuring Thing Points, enter the topic here and click Subscribe.

A screenshot of a web interface for subscribing to an IoT topic. The interface has a dark teal background. At the top left, the text "Subscribe:" is displayed. Below it is a white text input field containing the topic string "\$aws/things/myFirstThing/testTopic". To the right of the input field are two buttons: a blue "Subscribe" button and a grey "Unsubscribe" button. A mouse cursor is positioned over the "Subscribe" button.

Once you have subscribed to a topic, you can test the connection using the AWS IoT Test Client. Anything successfully sent by the test client to this topic will be displayed in the message window when Last Sub is clicked. Be sure to click Unsubscribe when done testing.



14. Configuring IoT Client to Publish to AWS

The MQTT term "Publish", from a controls perspective, would be most closely associated with the action of a sensor. You have data available that you wish to transmit to other devices or systems. In the instance we are working with here, we are Publishing data to the AWS server.

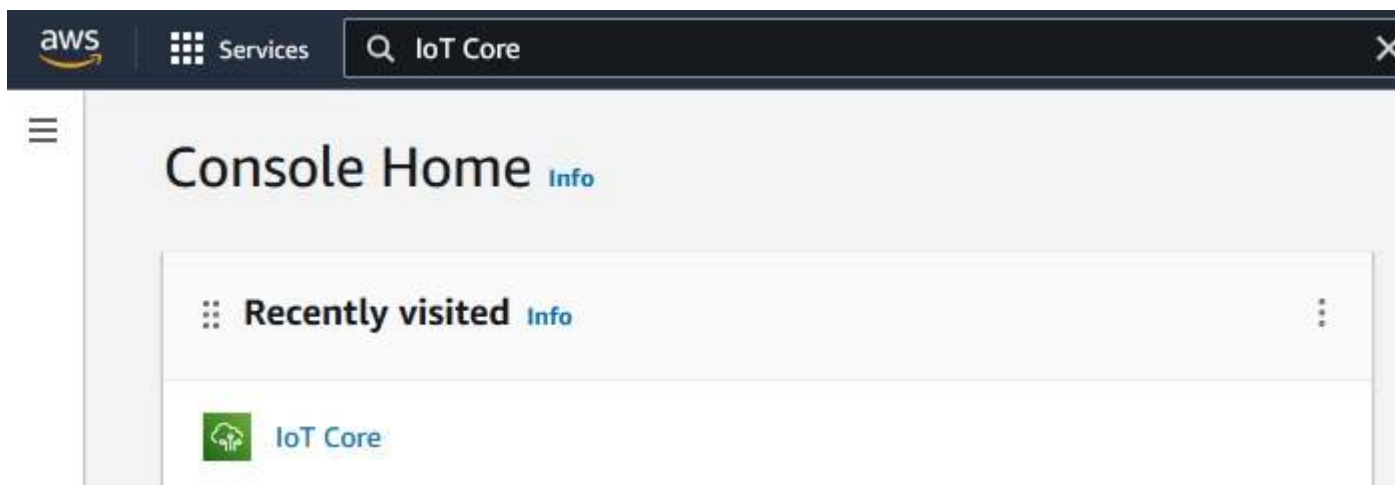
Configuring the ValuPoint to publish data to the Amazon servers requires setting things up on both ends: You need to configure the ValuPoint, and you need to configure your AWS account at Amazon.

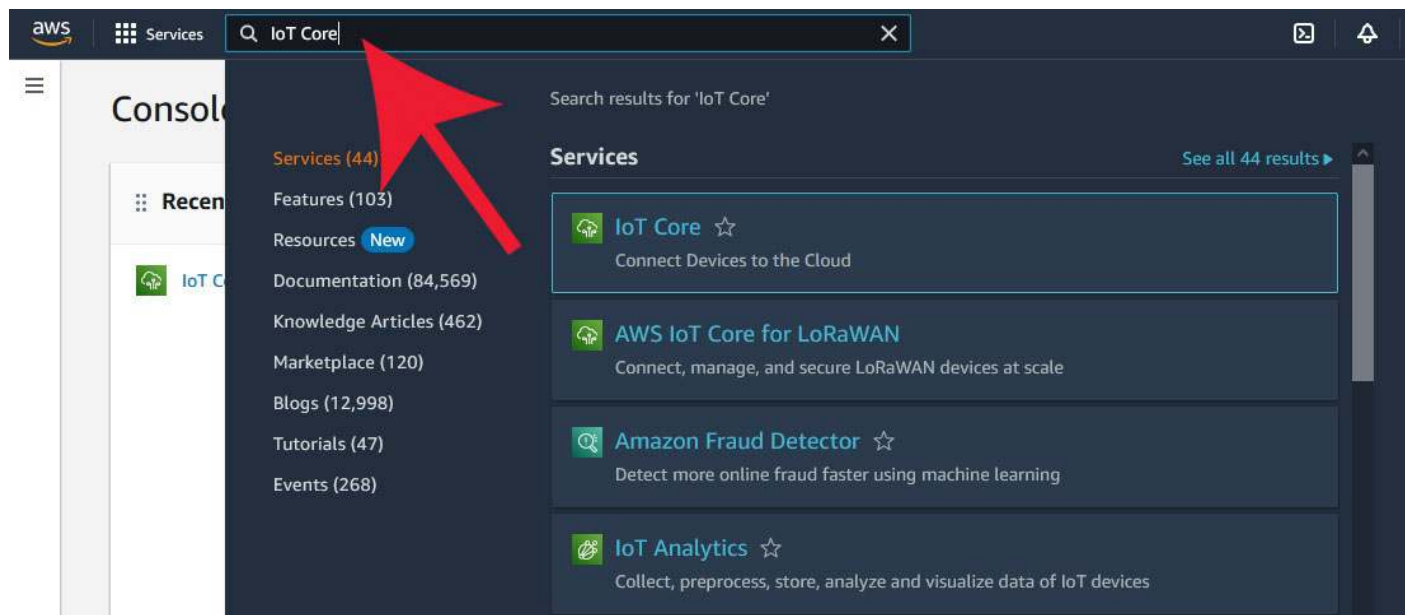
IMPORTANT: The screen shots illustrated in this document were Amazon's web interface as of when these were first captured. We are aware that a few months later, the appearance of some of these screens has changed, and Amazon is noting that there are more changes to come. They will never stop changing ("improving") how the pages look, but what you need to do and the general flow of how to do it have not changed, and probably won't for a long time.

14.1 Create and Register a "Thing"

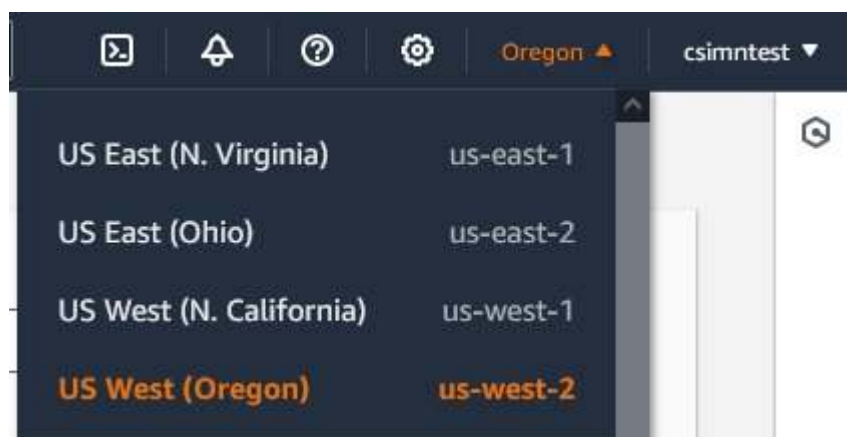
Start by going to <https://aws.amazon.com> and creating an account if you don't already have one. There is no fee to set up the account, and your data usage will also generally be free for an introductory period.

From your AWS Management Console, search for IoT Core or click on the IoT Core link under recently visited services.

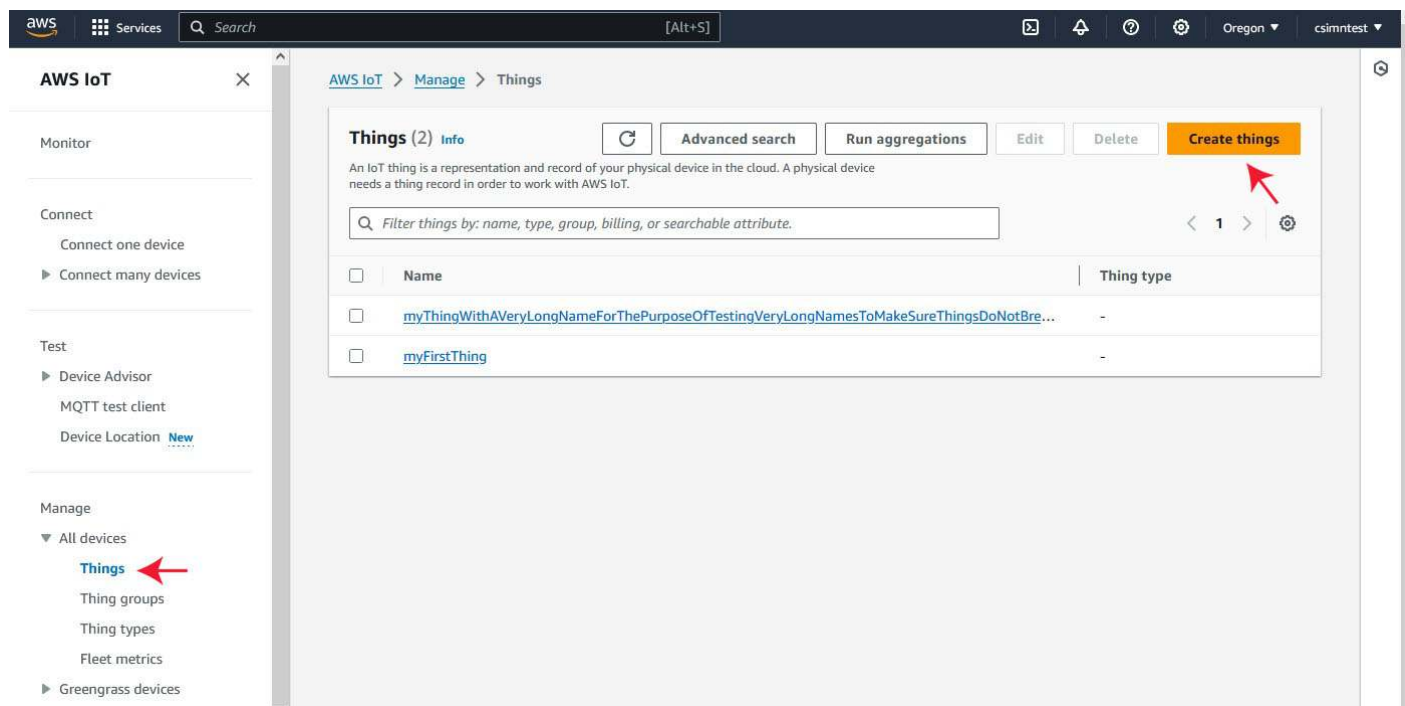




You will have a choice of regions in which to set up your Things. The abbreviated list is illustrated below. The actual list is much longer and includes all continents and numerous countries. If you have previously configured some Things, and see a message on the screen saying you have no things, check the region. Once in a while, the AWS login forgets what region you were working in.



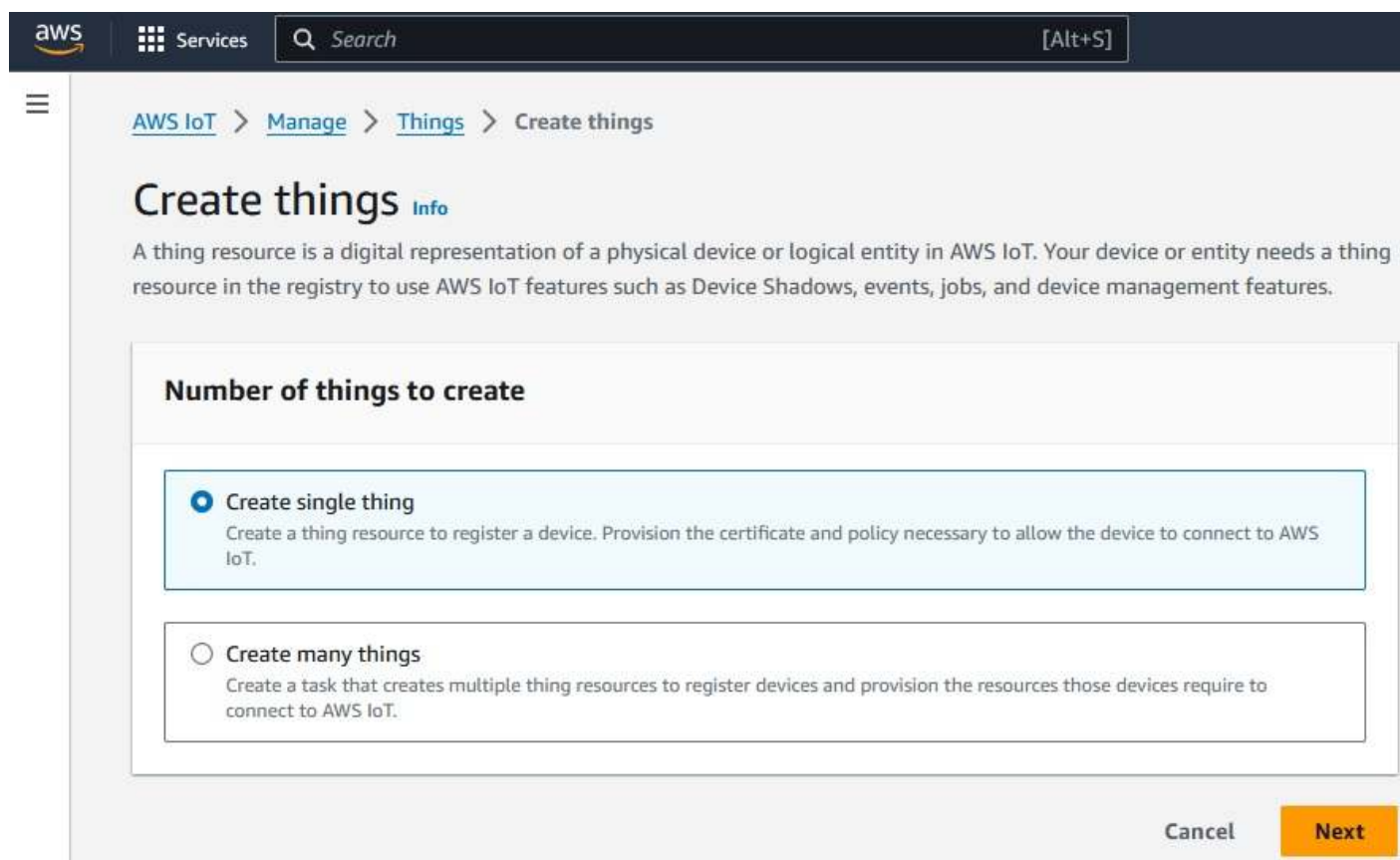
From the AWS IoT menu, click on Manage, and then select Things. You may initially get a screen telling you that you have no things. The screen shot here shows that we already have a couple of things created previously. Either way, click on the Create Things button.



The screenshot shows the AWS IoT console interface. On the left, there is a navigation menu with sections: Monitor, Connect, Test, and Manage. Under 'Manage', 'Things' is selected and highlighted with a red arrow. The main content area shows the 'Things (2)' page. At the top right, the 'Create things' button is highlighted with a red arrow. Below this, there is a search filter and a table with two rows of IoT things.

<input type="checkbox"/>	Name	Thing type
<input type="checkbox"/>	myThingWithAVeryLongNameForThePurposeOfTestingVeryLongNamesToMakeSureThingsDoNotBre...	-
<input type="checkbox"/>	myFirstThing	-

You have the option of creating a single Thing or multiple Things at once. We are going to step through creating just a single Thing here.



The screenshot shows the 'Create things' wizard in the AWS IoT console. The breadcrumb trail is 'AWS IoT > Manage > Things > Create things'. The main heading is 'Create things Info'. Below this, there is a section titled 'Number of things to create' with two radio button options: 'Create single thing' (selected) and 'Create many things'. The 'Create single thing' option is described as: 'Create a thing resource to register a device. Provision the certificate and policy necessary to allow the device to connect to AWS IoT.' The 'Create many things' option is described as: 'Create a task that creates multiple thing resources to register devices and provision the resources those devices require to connect to AWS IoT.' At the bottom right, there are 'Cancel' and 'Next' buttons.

Select 'Create single thing' and click Next. Then provide a name for your new Thing.

aws Services [Alt+S]

AWS IoT > Manage > Things > Create things > Create single thing

Step 1
Specify thing properties

Step 2 - optional
Configure device certificate

Step 3 - optional
Attach policies to certificate

Specify thing properties [Info](#)

A thing resource is a digital representation of a physical device or logical entity in AWS IoT. Your device or entity needs a thing resource in the registry to use AWS IoT features such as Device Shadows, events, jobs, and device management features.

Thing properties [Info](#)

Thing name

Enter a unique name containing only: letters, numbers, hyphens, colons, or underscores. A thing name can't contain any spaces.

Additional configurations

You can use these configurations to add detail that can help you to organize, manage, and search your things.

- ▶ **Thing type** - optional
- ▶ **Searchable thing attributes** - optional
- ▶ **Thing groups** - optional
- ▶ **Billing group** - optional
- ▶ **Packages and versions** - optional

Device Shadow [Info](#)

In addition to giving the Thing a name, scroll down and select 'Named shadow'. Give the shadow a name. Giving the shadow the same name as the Thing itself makes it easier to remember. Click Next.

IMPORTANT: If your Thing has no shadow at all, the ValuPoint IoT Client will suspend indefinitely because the first thing it needs to do when connecting is retrieve the shadow.

Thing properties [Info](#)

Thing name

Enter a unique name containing only: letters, numbers, hyphens, colons, or underscores. A thing name can't contain any spaces.

Additional configurations

You can use these configurations to add detail that can help you to organize, manage, and search your things.

- ▶ **Thing type** - *optional*
- ▶ **Searchable thing attributes** - *optional*
- ▶ **Thing groups** - *optional*
- ▶ **Billing group** - *optional*
- ▶ **Packages and versions** - *optional*

Device Shadow [Info](#)

Device Shadows allow connected devices to sync states with AWS. You can also get, update, or delete the state information of this thing's shadow using either HTTPs or MQTT topics.

No shadow

Named shadow
Create multiple shadows with different names to manage access to properties, and logically group your devices properties.

Unnamed shadow (classic)
A thing can have only one unnamed shadow.

Shadow name

Enter a unique name that contains only: letters, hyphens, colons, or underscores. A shadow name cannot contain any spaces.

▶ **Edit shadow statement** - *optional*

The certificates step is VERY IMPROTANT. Select 'Auto-generate'. Then click Next.

The screenshot shows the AWS IoT console interface for the 'Configure device certificate' step. The breadcrumb trail is 'AWS IoT > Manage > Things > Create things > Create single thing'. The left sidebar shows three steps: 'Step 1: Specify thing properties', 'Step 2 - optional: Configure device certificate', and 'Step 3 - optional: Attach policies to certificate'. The main content area is titled 'Configure device certificate - optional' and includes a description: 'A device requires a certificate to connect to AWS IoT. You can choose how to register a certificate for your device now, or you can create and register a certificate for your device later. Your device won't be able to connect to AWS IoT until it has an active certificate with an appropriate policy.' Below this is a 'Device certificate' section with four radio button options: 'Auto-generate a new certificate (recommended)' (selected), 'Use my certificate', 'Upload CSR', and 'Skip creating a certificate at this time'. At the bottom right are 'Cancel', 'Previous', and 'Next' buttons.

The next step is to identify policies to attach to the certificate. A policy with a name derived from your Thing name will be created automatically, but you need to select it. Note that for our purposes, the policy is not really "optional".

The screenshot shows the AWS IoT console interface for the 'Attach policies to certificate' step. The breadcrumb trail is 'AWS IoT > Manage > Things > Create things > Create single thing'. The left sidebar shows three steps: 'Step 1: Specify thing properties', 'Step 2 - optional: Configure device certificate', and 'Step 3 - optional: Attach policies to certificate'. The main content area is titled 'Attach policies to certificate - optional' and includes a description: 'AWS IoT policies grant or deny access to AWS IoT resources. Attaching policies to the device certificate applies this access to the device.' Below this is a 'Policies (2)' section with a 'Create policy' button and a search box labeled 'Filter policies'. A list of policies is shown with checkboxes: 'Name', 'myFirstThing_policy', and 'aValuPoint-Policy'. At the bottom right are 'Cancel', 'Previous', and 'Create thing' buttons.

Select your named policy from the list, then click Create Thing. Don't worry about what the policy contains for now. You will update that after creating the Thing.

The screenshot shows the AWS IoT console interface. The breadcrumb navigation is: AWS IoT > Manage > Things > Create things > Create single thing. The current step is 'Step 3 - optional: Attach policies to certificate'. The main heading is 'Attach policies to certificate - optional' with an 'Info' link. Below the heading is a description: 'AWS IoT policies grant or deny access to AWS IoT resources. Attaching policies to the device certificate applies this access to the device.'

The 'Policies (1/2)' section contains a search bar with the text 'Filter policies' and a refresh button. Below the search bar is a table with the following content:

	Name
<input type="checkbox"/>	myFirstThing_policy
<input checked="" type="checkbox"/>	aValuPoint-Policy

At the bottom of the console, there are three buttons: 'Cancel', 'Previous', and 'Create thing'.

PAY ATTENTION here to the fact that this is the ONLY opportunity you will get to download your SSL certificate key files. If you miss this step, you will need to delete your Thing and start over.

You need to download 3 files: Device certificate, private key, and root CA. The public key file will not be used in the ValuPoint but you can download it anyway.

Download certificates and keys ✕

Download certificate and key files to install on your device so that it can connect to AWS.


Device certificate

You can activate the certificate now, or later. The certificate must be active for a device to connect to AWS IoT.

Device certificate f2207cfd6b0...te.pem.crt	<button>Deactivate certificate</button>	<button>Download</button>
--	---	---------------------------

Key files

The key files are unique to this certificate and can't be downloaded after you leave this page. Download them now and save them in a secure place.


 This is the only time you can download the key files for this certificate.

Public key file f2207cfd6b098ad7109b6d5...d7df89f-public.pem.key	<button>Download</button>
Private key file f2207cfd6b098ad7109b6d5...7df89f-private.pem.key	<button>Download</button>

Root CA certificates

Download the root CA certificate file that corresponds to the type of data endpoint and cipher suite you're using. You can also download the root CA certificates later.

Amazon trust services endpoint RSA 2048 bit key: Amazon Root CA 1	<button>Download</button>
Amazon trust services endpoint ECC 256 bit key: Amazon Root CA 3	<button>Download</button>

If you don't see the root CA certificate that you need here, AWS IoT supports additional root CA certificates. These root CA certificates and others are available in our developer guides. [Learn more](#) 

Done

Once you have downloaded your files, it will indicate that fact on the screen. Once you have downloaded all 3 files, you may click Done.

Download certificates and keys ✕

Download certificate and key files to install on your device so that it can connect to AWS.


Device certificate

You can activate the certificate now, or later. The certificate must be active for a device to connect to AWS IoT.

Device certificate f2207cfd6b0...te.pem.crt	<button>Deactivate certificate</button>	<button>Download</button>
--	---	---------------------------

Key files

The key files are unique to this certificate and can't be downloaded after you leave this page. Download them now and save them in a secure place.

 This is the only time you can download the key files for this certificate.


Public key file f2207cfd6b098ad7109b6d5...d7df89f-public.pem.key	<button>Download</button>
Private key file f2207cfd6b098ad7109b6d5...7df89f-private.pem.key	<button>Download</button>

✓ Key downloaded
✓ Key downloaded

Root CA certificates

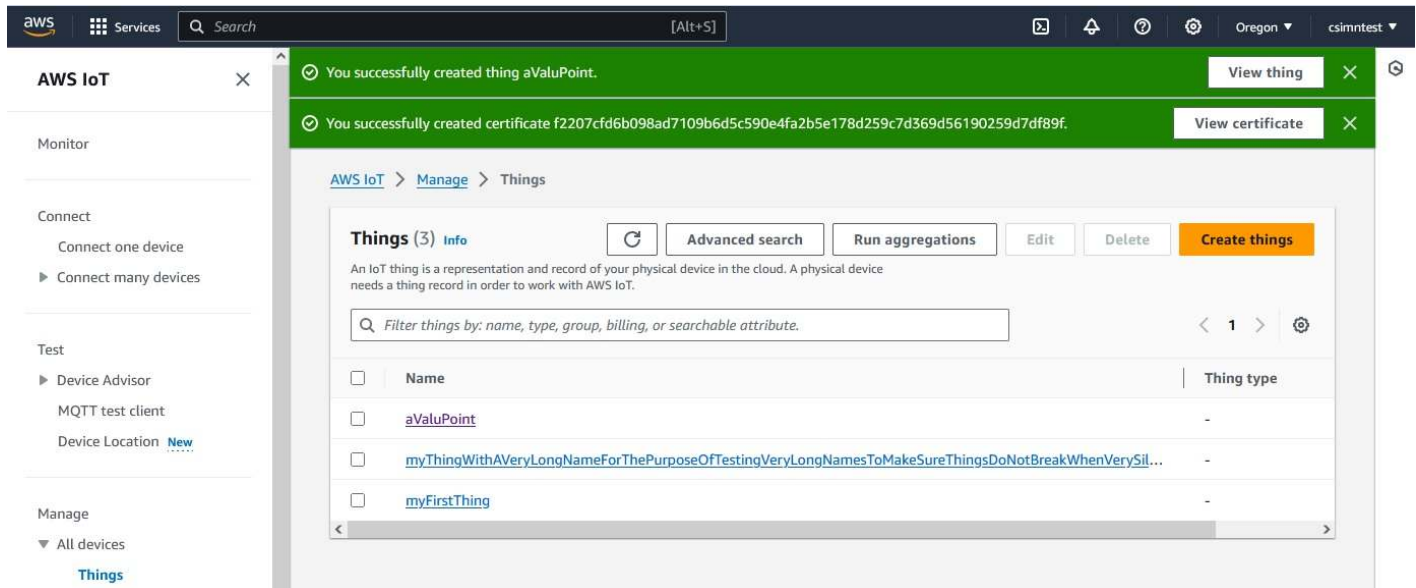
Download the root CA certificate file that corresponds to the type of data endpoint and cipher suite you're using. You can also download the root CA certificates later.

Amazon trust services endpoint RSA 2048 bit key: Amazon Root CA 1	<button>Download</button>
Amazon trust services endpoint ECC 256 bit key: Amazon Root CA 3	<button>Download</button>

If you don't see the root CA certificate that you need here, AWS IoT supports additional root CA certificates. These root CA certificates and others are available in our developer guides. [Learn more](#) 

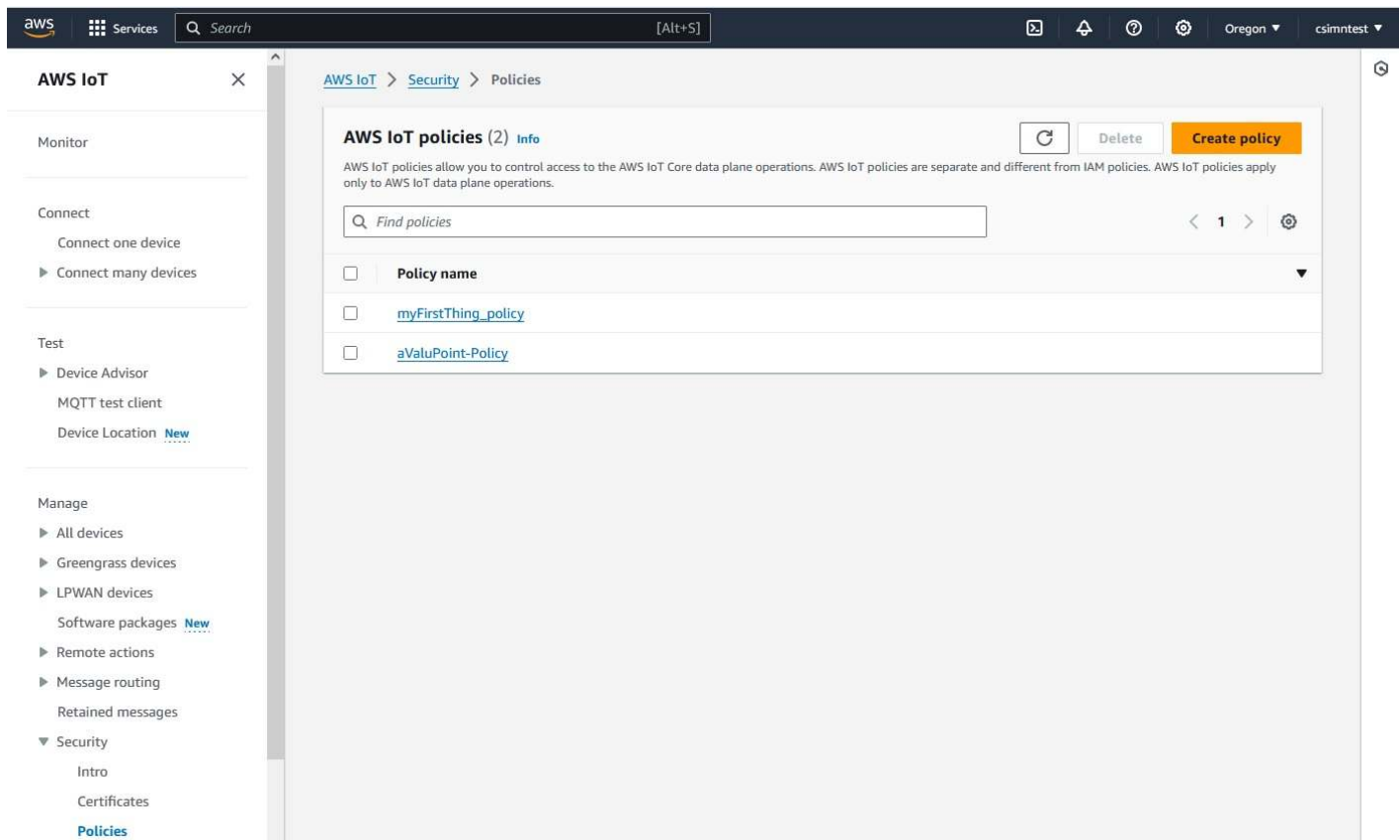
Done

The newly created Thing now shows up on your Things list.



14.2 Update Policy

You can find your security policies under Security.



Click on the policy that was attached to the Thing you just created. It will default to looking something like the following screen shot.

Details

Policy ARN arn:aws:iot:us-west-2:314429653841:policy/aValuPoint-Policy	Active version 1	Created March 04, 2024, 10:33:45 (UTC-06:00)	Last updated March 04, 2024, 10:33:45 (UTC-06:00)
---	---------------------	---	--

Active version: 1

Policy effect	Policy action	Policy resource
Allow	iot:Publish	arn:aws:iot:us-west-2:314429653841:topic/sdk/test/java
Allow	iot:Publish	arn:aws:iot:us-west-2:314429653841:topic/sdk/test/python
Allow	iot:Publish	arn:aws:iot:us-west-2:314429653841:topic/sdk/test/js
Allow	iot:Receive	arn:aws:iot:us-west-2:314429653841:topic/sdk/test/java
Allow	iot:Receive	arn:aws:iot:us-west-2:314429653841:topic/sdk/test/python
Allow	iot:Receive	arn:aws:iot:us-west-2:314429653841:topic/sdk/test/js
Allow	iot:PublishRetain	arn:aws:iot:us-west-2:314429653841:topic/sdk/test/java
Allow	iot:PublishRetain	arn:aws:iot:us-west-2:314429653841:topic/sdk/test/python

Click on 'Edit active version'. The Builder view of the policy will look something like the following screen shot.

Edit policy: aValuPoint-Policy (Version 1)

Policy document

An AWS IoT policy contains one or more policy statements. Each policy statement contains actions, resources, and an effect that grants or denies the actions by the resources.

Policy effect	Policy action	Policy resource	
Allow	iot:Publish,iot:Receive,iot:Publish	arn:aws:iot:us-west-2:314429653841:topic/sdk/test/java	Remove
Allow	iot:Subscribe	arn:aws:iot:us-west-2:314429653841:topic/sdk/test/python	Remove
Allow	iot:Connect	arn:aws:iot:us-west-2:314429653841:topic/sdk/test/js	Remove

Add new statement

Switch to the JSON view. It will look something like the following. This content is created automatically by AWS with their assuming you are going to connect to AWS from an app on your PC. That is not the case for your ValuPoint IoT device, so the content of the policy needs to be replaced.

[AWS IoT](#) > [Security](#) > [Policies](#) > [aValuPoint-Policy](#) > Edit

Edit policy: aValuPoint-Policy (Version 1)

[Policy statements](#)[Policy examples](#)

Policy document [Info](#)

Builder

JSON

An AWS IoT policy contains one or more policy statements. Each policy statement contains actions, resources, and an effect that grants or denies the actions by the resources.

Policy document

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": [
7         "iot:Publish",
8         "iot:Receive",
9         "iot:PublishRetain"
10      ],
11      "Resource": [
12        "arn:aws:iot:us-west-2:314429653841:topic/sdk/test/java",
13        "arn:aws:iot:us-west-2:314429653841:topic/sdk/test/python",
14        "arn:aws:iot:us-west-2:314429653841:topic/sdk/test/js"
15      ]
16    },
17    {
18      "Effect": "Allow",
19      "Action": "iot:Subscribe",
20      "Resource": [
21        "arn:aws:iot:us-west-2:314429653841:topicfilter/sdk/test/java",
22        "arn:aws:iot:us-west-2:314429653841:topicfilter/sdk/test/python",
23        "arn:aws:iot:us-west-2:314429653841:topicfilter/sdk/test/js"
24      ]
25    }
26  ]
27 }
```

JSON

Ln 1, Col 1

Errors: 0

Warnings: 0



The content you want to start with at least for test purposes is illustrated in the screen shot below. Once you get your ValuPoint talking, you can come back later and tighten the security after you familiarize yourself with policies per AWS documentation.

Policy document [Info](#)

Builder JSON

An AWS IoT policy contains one or more policy statements. Each policy statement contains actions, resources, and an effect that grants or denies the actions by the resources.

Policy document

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": [
7         "iot:Publish",
8         "iot:Receive",
9         "iot:PublishRetain"
10      ],
11      "Resource": "*"
12    },
13    {
14      "Effect": "Allow",
15      "Action": "iot:Subscribe",
16      "Resource": "*"
17    },
18    {
19      "Effect": "Allow",
20      "Action": "iot:Connect",
21      "Resource": "*"
22    }
23  ]
24 }
```

JSON Ln 24, Col 2 ✖ Errors: 0 ⚠ Warnings: 0

Policy version status

Active policy

Set the edited version as the active version for this policy

You can change this setting later in the policy's detail page.

Cancel Save as new version

This policy in text form that you can copy from this document and paste into AWS is as follows:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Receive",
        "iot:PublishRetain"
      ],
      "Resource": "*"
    },
  ],
}
```

```

{
  "Effect": "Allow",
  "Action": "iot:Subscribe",
  "Resource": "*"
},
{
  "Effect": "Allow",
  "Action": "iot:Connect",
  "Resource": "*"
}
]
}

```

Once updated, the Builder view will look more like the following screen shot.

✔ Successfully updated policy aValuPoint-Policy, creating policy version 2.
✕

[AWS IoT](#) > [Security](#) > [Policies](#) > aValuPoint-Policy

aValuPoint-Policy Info

Edit active version Delete

Details

Policy ARN <code>arn:aws:iot:us-west-2:314429653841:policy/aValuPoint-Policy</code>	Active version 2	Created March 04, 2024, 10:33:45 (UTC-06:00)	Last updated March 04, 2024, 10:33:45 (UTC-06:00)
--	---------------------	---	--

[Versions](#) | [Targets](#) | [Noncompliance](#) | [Tags](#)

Active version: 2 Info Builder JSON

Policy effect	Policy action	Policy resource
Allow	iot:Publish	*
Allow	iot:Receive	*
Allow	iot:PublishRetain	*
Allow	iot:Subscribe	*
Allow	iot:Connect	*

All versions (2) Info Refresh Delete Set as active Edit version View JSON

The active and previous versions of this policy. Only one version can be active. A policy can have no more than 5 versions. To update a policy with 5 versions, you must first delete one.

<input type="checkbox"/>	Version number	Status	Created
<input type="checkbox"/>	2	✔ Active	March 04, 2024, 11:09:08 (UTC-06:00)

14.3 Configure ValuPoint as a “Thing”

Copy your AWS host name from the Settings page as illustrated below.

The screenshot shows the AWS IoT Settings page. The left sidebar has a 'Settings' link highlighted with a red arrow. The main content area shows the 'Device data endpoint' section with the endpoint URL 'a1lzj6a7dtjmhf-ats.iot.us-west-2.amazonaws.com' highlighted by another red arrow. Below this is the 'Domain configurations' section, which is currently empty.

Paste this host name into the Server Host Name window on the Thing ID page in your ValuPoint. Provide the Thing name you created. Use server port 8883, and do not disable SSL

The screenshot shows the ValuPoint 'Thing ID' configuration page. The 'Server Host Name' field contains 'a1lzj6a7dtjmhf-ats.iot.us-west-2.amazonaws.com'. The 'Server Port' field contains '8883'. There are checkboxes for 'Disable SSL' and 'Disable SSL certificate verify', both of which are unchecked. The 'Thing Name / Client ID' field contains 'aValuPoint'. An 'Update' button is visible in the top right corner.

The certificates you downloaded will have very long names looking something like the following. The 3 files highlighted here are the ones you will need to upload to your ValuPoint.

File Name	Date	Type	Size
AmazonRootCA1.pem	3/4/2024 10:54 AM	PEM File	2 KB
f2207cfd6b098ad7109b6d5c590e4fa2b5e178d259c7d369d56190259d7df89f-certificate.pem.crt	3/4/2024 10:53 AM	Security Certificate	2 KB
f2207cfd6b098ad7109b6d5c590e4fa2b5e178d259c7d369d56190259d7df89f-private.pem.key	3/4/2024 10:54 AM	KEY File	2 KB
f2207cfd6b098ad7109b6d5c590e4fa2b5e178d259c7d369d56190259d7df89f-public.pem.key	3/4/2024 10:54 AM	KEY File	1 KB
upload	3/4/2024 11:17 AM	File folder	

Rename the files with shorter names, and make sure the suffix is .pem, not key or crt. The content of all of these files is in pem format. The renamed files should look more like the following.

ValuPoint Certs > upload Search upload

Name	Date modified	Type	Size
AmazonRootCA1.pem	3/4/2024 10:54 AM	PEM File	2 KB
f2207cf6b098-certificate.pem	3/4/2024 10:53 AM	PEM File	2 KB
f2207cf6b098-private.pem	3/4/2024 10:54 AM	PEM File	2 KB

Go to the File Manager page in the ValuPoint and upload all 3 files.

File Manager | Network | Resources | User | I/O Config

Free space: 1.35 MB

File Directory: AmazonRootCA1.pem

Filtered by: *.pem | Filter | View | Select

Selected File: AmazonRootCA1.pem

Selected File: f2207cf6b098-certificate.pem

Selected File: f2207cf6b098-private.pem

Action: --- select --- | Execute

Boot configuration: Test2.xml

Confirm | Restart

Upload File

Upload | Choose File | No file chosen

Now go to the Thing Files page, and apply the 3 files. Select a file from the list, and click the applicable Apply button.

Local Data | Modbus | IoT Cloud | System

Thing Setup | Thing Status

Thing Points | Thing ID | **Thing Files**

Local file directory: AmazonRootCA1.pem | View

Apply Device Cert

Apply Private Key



Apply Root CA | AmazonRootCA1.pem

Once all 3 files are applied, the page in the ValuPoint will look something like this. These file names along with the rest of your Thing configuration in the ValuPoint are stored in the XML configuration file. After setting up the Thing, be sure to go to the File Manager page and Save your configuration to the XML file that will be reloaded at power up or restart.

Thing Points	Thing ID	Thing Files
Local file directory: f2207cfd6b098-private.pem <input type="button" value="View"/>		
Apply Device Cert	f2207cfd6b098-certificate.pem	
Apply Private Key	f2207cfd6b098-private.pem	
Apply Root CA	AmazonRootCA1.pem	

Once your Thing has been created both at AWS and in the ValuPoint (and assuming you have defined some thing points as outlined in Section 13), you can proceed to enable the IoT Engine.

Do note that the features enabled for an AWS connection should include AWS IoT Core and Complex JSON.

Local Data	Modbus	IoT Cloud	System
Thing Setup	Thing Status		
Thing Points	Thing ID	Thing Files	Thing ID

Server Host Name: a1lzj6a7dtjmhf-ats.iot.us-west-2.amazonaws.com

Server Port: 8883 Disable SSL Disable SSL certificate verify

Thing Name / Client ID: aValuPoint

Username:

Password:

Features Enabled: AWS IoT Core Complex JSON Thingsboard RPC

IoT Engine Status: Enabled (See IMPORTANT Note Below)

Subscribe Topics:

Topic 0: \$aws/things/aValuPoint/shadow/update

Topic 1:

Topic 2:

Topic 3:

Topic 4:

14.4 Test Publish to Device Shadow

You can verify that your ValuPoint connected to AWS on the Connection page.

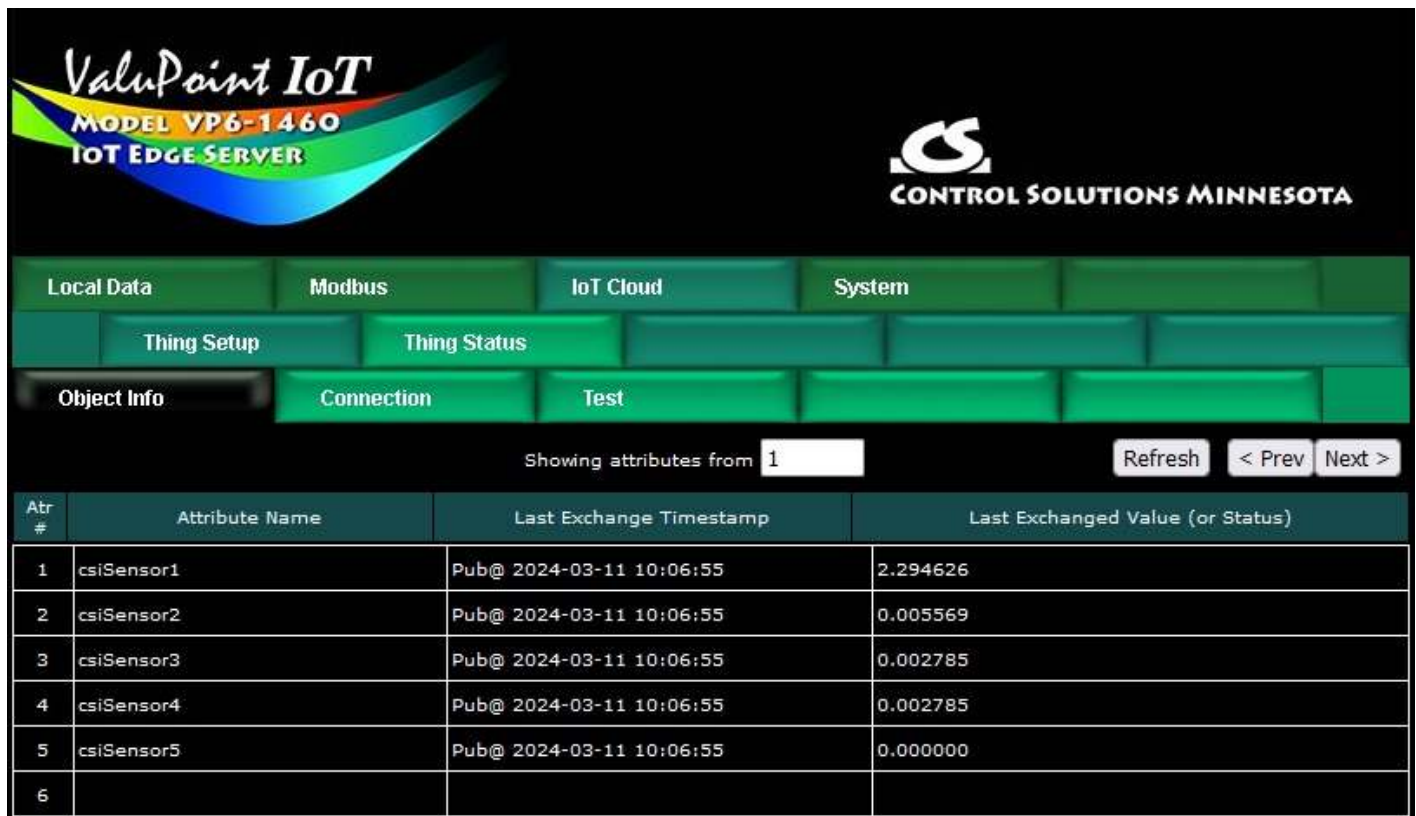


The screenshot displays the ValuPoint IoT Edge Server web interface. The top left corner features the ValuPoint IoT logo with the text "MODEL VP6-1460 IOT EDGE SERVER". The top right corner shows the logo for CONTROL SOLUTIONS MINNESOTA. The interface has a navigation menu with tabs for "Local Data", "Modbus", "IoT Cloud", and "System". Under "IoT Cloud", there are sub-tabs for "Thing Setup" and "Thing Status". Under "Thing Status", there are sub-tabs for "Object Info", "Connection", and "Test". The "Connection" tab is currently selected. In the top right corner of the main content area, there are "Clear" and "Refresh" buttons. The main content area shows the following information:

Connection Status	Connect: Success
Failed Connection Count	0
Publish Message Count	0
Publish Error Count	0
Subscribe Message Count	0
Subscribe Error Count	0

Below the table, there is a "Connection Info" section with the text: "Connecting to server at a1lj6a7dtjmhf-ats.iot.us-west-2.amazonaws.com:8883. Connected securely!"

If you are connected successfully, do whatever is applicable per your Thing Point rules to cause it to publish to AWS. Once something has been published, the Object Info page will show a timestamp and value sent to AWS.



ValuPoint IoT
MODEL VP6-1460
IOT EDGE SERVER

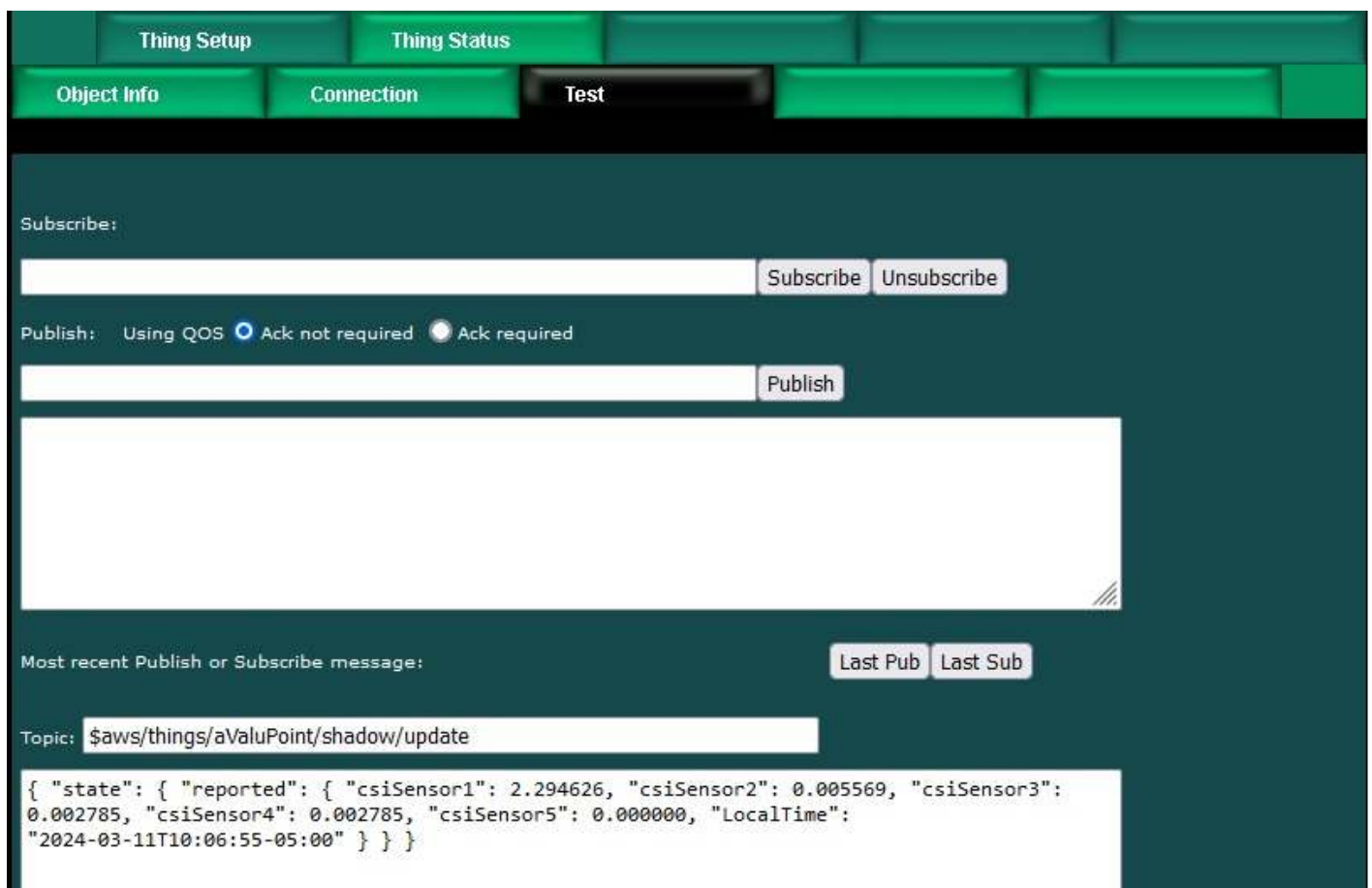
CONTROL SOLUTIONS MINNESOTA

Local Data Modbus IoT Cloud System
Thing Setup Thing Status
Object Info Connection Test

Showing attributes from 1 Refresh < Prev Next >

Atr #	Attribute Name	Last Exchange Timestamp	Last Exchanged Value (or Status)
1	csiSensor1	Pub@ 2024-03-11 10:06:55	2.294626
2	csiSensor2	Pub@ 2024-03-11 10:06:55	0.005569
3	csiSensor3	Pub@ 2024-03-11 10:06:55	0.002785
4	csiSensor4	Pub@ 2024-03-11 10:06:55	0.002785
5	csiSensor5	Pub@ 2024-03-11 10:06:55	0.000000
6			

You can also go to the Test page and click the Last Pub and Last Sub buttons to see the most recent transaction in raw JSON format.



Thing Setup Thing Status
Object Info Connection Test

Subscribe:
 Subscribe Unsubscribe

Publish: Using QOS Ack not required Ack required
 Publish

Most recent Publish or Subscribe message: Last Pub Last Sub

Topic: \$aws/things/aValuPoint/shadow/update

```
{ "state": { "reported": { "csiSensor1": 2.294626, "csiSensor2": 0.005569, "csiSensor3": 0.002785, "csiSensor4": 0.002785, "csiSensor5": 0.000000, "LocalTime": "2024-03-11T10:06:55-05:00" } } }
```


Back on the AWS host side, under details for your Thing, you can look at your device shadow.

The screenshot shows the AWS IoT console interface for a device named 'aValuPoint'. The breadcrumb navigation is 'AWS IoT > Manage > Things > aValuPoint'. The device name 'aValuPoint' is displayed with an 'Info' link. Action buttons for 'Create secure tunnel', 'Edit', and 'Delete' are visible. The 'Thing details' section shows the following information:

Name	aValuPoint	Type	-
ARN	arn:aws:iot:us-west-2:314429653841:thing/aValuPoint	Billing group	-

Below the details is a navigation bar with tabs: 'Attributes', 'Certificates', 'Thing groups', 'Device Shadows' (selected), 'Activity', and 'Packages and ve'. The 'Device Shadows' section shows 'Device Shadows (1) Info' with a refresh button, a 'Delete' button, and a 'Create Device Shadow' button. A descriptive text states: 'Device Shadows allow connected devices to sync their state with AWS. You can also get, update or delete the state information about this thing's Device Shadows by using HTTPS and MQTT topics.' There is a search box labeled 'Filter Device Shadows' and a pagination control showing '1' items. A table lists the device shadow:

<input type="checkbox"/>	Name ▲	MQTT topic prefix	Fleet indexing status	Loc
<input type="checkbox"/>	Classic Sh...	\$aws/things/aValuPoint/shadow	⊖ Not selected	⊖

Click on the shadow name to see the content of the shadow. As illustrated in the example below, you should see the data most recently published by your ValuPoint.

Device Shadow document [Info](#)

[Edit](#)

The Device Shadow document contains the reported, desired, and delta values of the device's state. You can edit the state values here or programmatically. Your device can sync its state while it's connected to AWS IoT.

Device Shadow state

```
{
  "state": {
    "desired": {
      "welcome": "aws-iot"
    },
    "reported": {
      "welcome": "aws-iot",
      "csiSensor1": 2.294626,
      "csiSensor2": 0.005569,
      "csiSensor3": 0.002785,
      "csiSensor4": 0.002785,
      "csiSensor5": 0.0,
      "LocalTime": "2024-03-11T10:06:55-05:00"
    }
  }
}
```



15. Configuring IoT Client to Subscribe to AWS

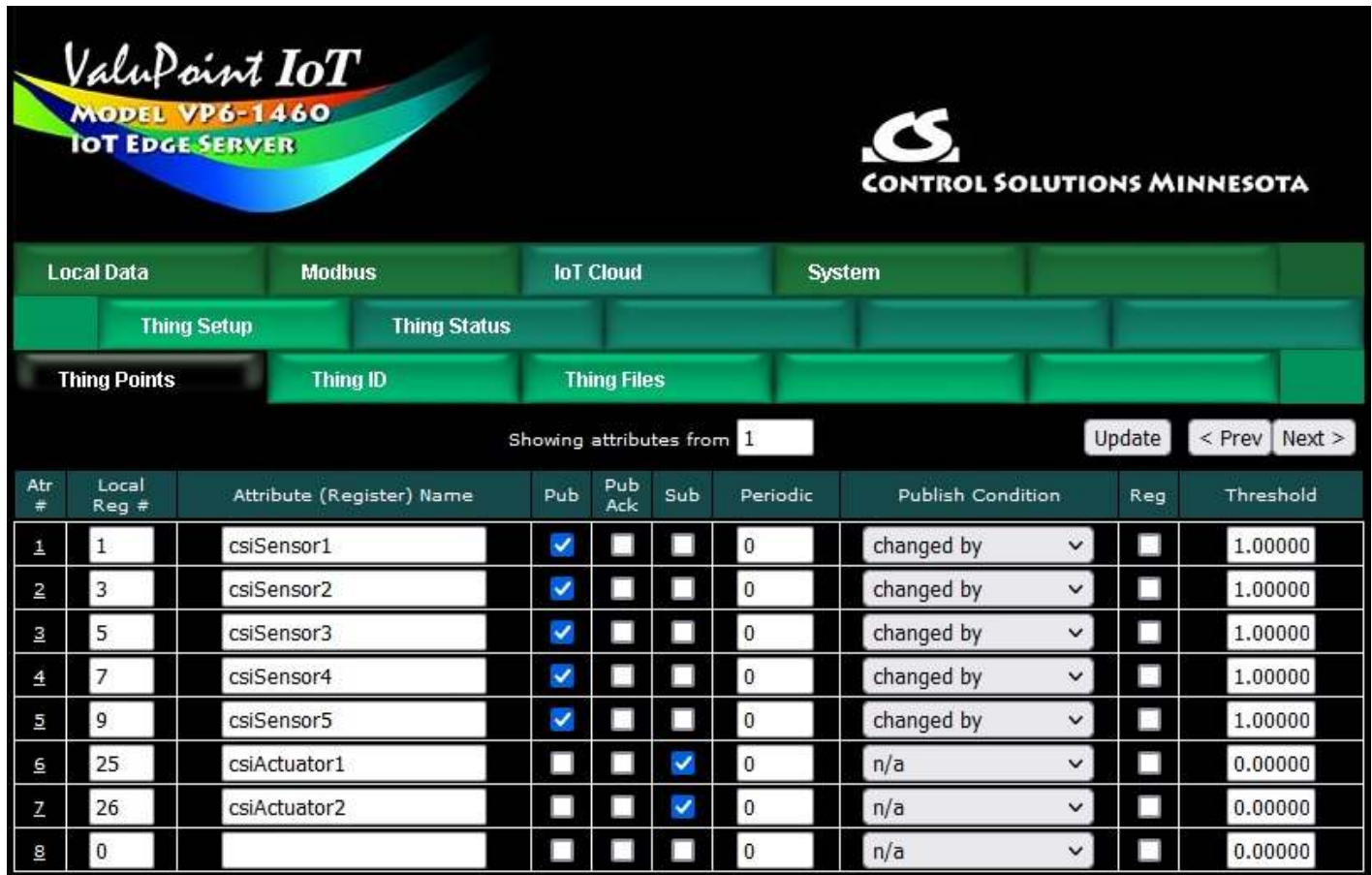
The MQTT term "Subscribe", from a controls perspective, would be most closely associated with the action of an actuator. You want to control an output based on setpoint data received from some external source. In the instance we are working with here, we are Subscribing to data from the AWS server.

The ValuPoint is programmed to look for the "desired" state for attributes. When some other device publishes to this device's Shadow using the "desired" state, this device will pick that up through its subscription, and in the case of the ValuPoint, write the received data value to a local register. That local register may then be subsequently written to a Modbus register in some other Modbus device, or be applied to local physical I/O.

While using the Shadow object is not required, the benefit of doing so is that the desired value will still be available should the ValuPoint be temporarily offline or disconnected. When the ValuPoint reconnects, it will retrieve the desired state information from the Shadow object and set its local registers accordingly.

15.1 Configure IoT Client

The simple form of a Subscribe point is illustrated below. To see the expanded view, click on the attribute number in the first column.



ValuPoint IoT
MODEL VP6-1460
IOT EDGE SERVER

CONTROL SOLUTIONS MINNESOTA

Local Data Modbus IoT Cloud System

Thing Setup Thing Status

Thing Points Thing ID Thing Files

Showing attributes from 1 Update < Prev Next >

Atr #	Local Reg #	Attribute (Register) Name	Pub	Pub Ack	Sub	Periodic	Publish Condition	Reg	Threshold
1	1	csiSensor1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by	<input type="checkbox"/>	1.00000
2	3	csiSensor2	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by	<input type="checkbox"/>	1.00000
3	5	csiSensor3	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by	<input type="checkbox"/>	1.00000
4	7	csiSensor4	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by	<input type="checkbox"/>	1.00000
5	9	csiSensor5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by	<input type="checkbox"/>	1.00000
6	25	csiActuator1	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	0	n/a	<input type="checkbox"/>	0.00000
7	26	csiActuator2	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	0	n/a	<input type="checkbox"/>	0.00000
8	0		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	n/a	<input type="checkbox"/>	0.00000

You will most often just use the default subscribe topic (illustrated below for "aValuPoint"), so all you really need to do is select Subscribe using the checkbox. Then click Update.

ValuPoint IoT
MODEL VP6-1460
IoT EDGE SERVER

CONTROL SOLUTIONS MINNESOTA

Local Data | Modbus | IoT Cloud | System

Thing Setup | Thing Status

Thing Points | Thing ID | Thing Files

Attribute # 6 [Update] < Prev Next >

Associate local register # 25 named csiActuator1 with this IoT attribute.

Publish: Using QOS Ack not required Ack required Publish as Reported Desired

MQTT Topic: Default Other \$aws/things/aValuPoint/shadow/update

Publish if register value is n/a this value: 0.000000 this local register: 0

Qualified by this hysteresis value: 0.000000 this minimum On Time: 0:00:00 this minimum Off Time: 0:00:00

Publish at least every 0 minutes. Publish no more than every 0 minutes.

Follow above rule only if local register 0 is set to a value of 0

Publish message on true: _____

Publish message on false: _____

Publish as part of dataset number: 0 Include timestamp

Subscribe: To topic index: 0 \$aws/things/aValuPoint/shadow/update

Apply this default value: 0.000000 after 0 minutes without any update from the cloud.

Attributes Enabled: 6 [Make Template] [Force Publish] [Insert] [Delete]

The default topic will normally be created for you when you enter the thing name.

Thing Points | **Thing ID** | Thing Files | Update

Server Host Name: a1lzj6a7dtjmhf-ats.iot.us-west-2.amazonaws.com

Server Port: 8883 Disable SSL Disable SSL certificate verify

Thing Name / Client ID: aValuPoint

Username:

Password:

Features Enabled: AWS IoT Core Complex JSON Thingsboard RPC

IoT Engine Status: Enabled (See IMPORTANT Note Below)

Subscribe Topics:

Topic 0: \$aws/things/aValuPoint/shadow/update

Topic 1:

15.2 Use MQTT Test Client to Test Subscription

Once your Thing Point is set up to Subscribe, go back to the AWS IoT management console and locate the MQTT test client. Using the MQTT client found here, you will manually publish to our default topic.

AWS IoT > Manage > Things > aValuPoint

aValuPoint Info

Create secure tunnel Edit Delete

Thing details

Name	aValuPoint	Type	-
ARN	arn:aws:iot:us-west-2:314429653841:thing/aValuPoint	Billing group	-

Device Shadows Activity Packages and versions Jobs Alarms Defender metrics

Activity (0) Info

Clear MQTT test client

Lists the most recent MQTT messages related to Device Shadow activity since you opened the thing details page. To see more messages related to this activity, choose the **MQTT test client** button.

No messages

Activity messages for your thing will show here when they are published.

You may want to use a generic text editor to prepare your JSON message ahead of time. The format illustrated below must be used. If you are interested in learning more about JSON and the Shadow update syntax, you can find more information both on the Amazon web site, and on the Internet in general.

Topic:

`$aws/things/aValuPoint/shadow/update`

"Nice" human readable JSON query:

```
{ "state":
  { "desired":
    { "csiActuator1": 1 }
  }
}
```

Single line JSON query equally acceptable to MQTT:

```
{ "state": { "desired": { "csiActuator1": 1 } } }
```

[AWS IoT](#) > MQTT test client

MQTT test client [Info](#)

You can use the MQTT test client to monitor the MQTT messages being passed in your AWS account. Devices publish MQTT messages that are identified by topics to communicate their state to AWS IoT. AWS IoT also publishes MQTT messages to inform devices and apps of changes and events. You can subscribe to MQTT message topics and publish MQTT messages to topics by using the MQTT test client.

▶ Connection details

✔ Connected

You can update the connection details by choosing Disconnect and making updates on the Establish connection to continue page.

Subscribe to a topic

Publish to a topic

Topic name

The topic name identifies the message. The message payload will be published to this topic with a Quality of Service (QoS) of 0.



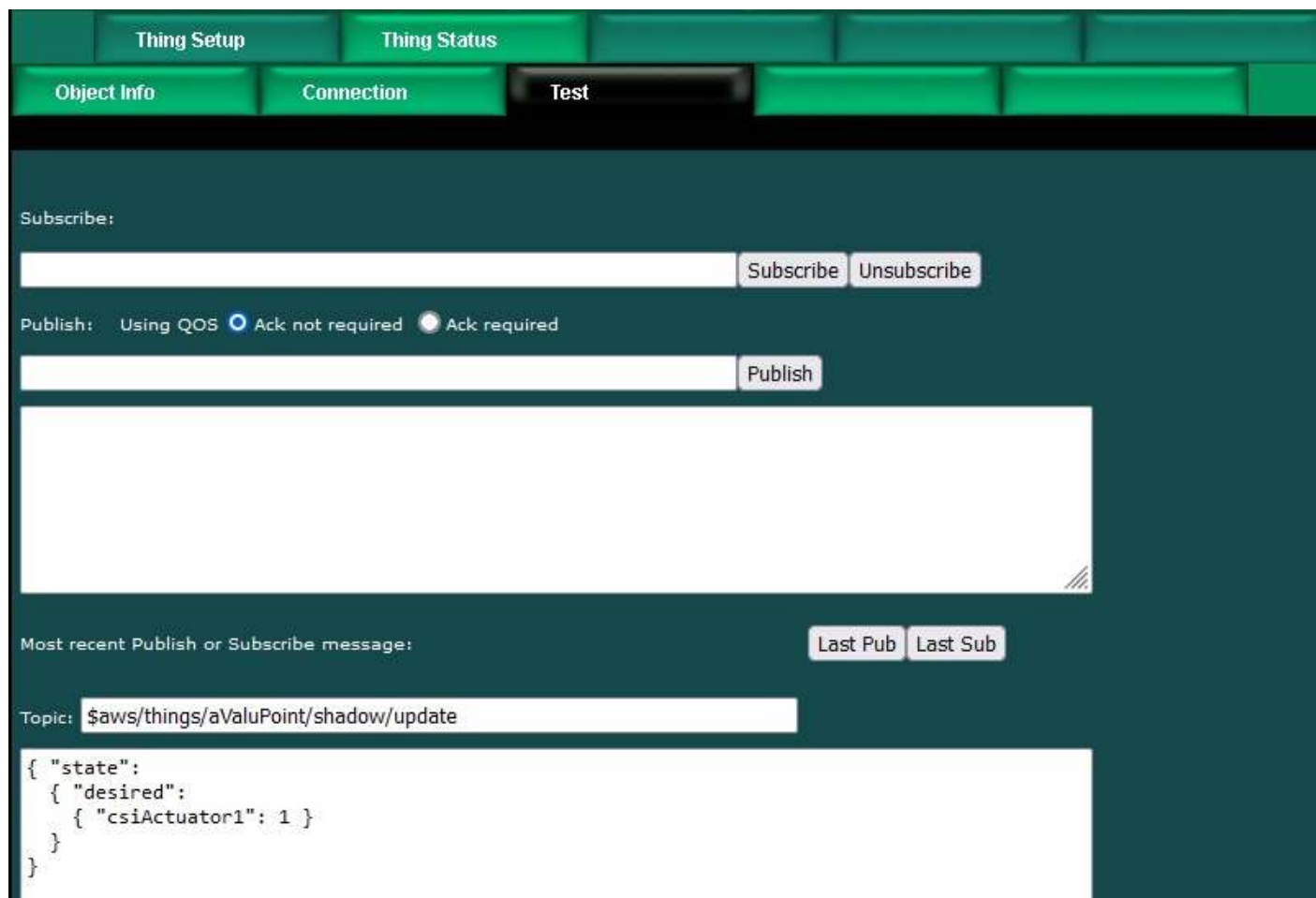
Message payload

```
{ "state":  
  { "desired":  
    { "csiActuator1": 1 }  
  }  
}
```

▶ Additional configuration

In the Publish section of the MQTT Client page, enter the topic illustrated above, and the payload. Then click Publish. If you go back and look at your thing's Shadow at this point you will see the "desired" state that was published.

On the ValuPoint Test page, when you now click Last Sub, you should see your message show up at the ValuPoint.



The screenshot displays a web-based interface for an IoT client. At the top, there are navigation tabs: "Thing Setup", "Thing Status", "Object Info", "Connection", and "Test". The "Test" tab is currently selected. Below the tabs, there are two main sections: "Subscribe:" and "Publish:".

The "Subscribe:" section includes a text input field, a "Subscribe" button, and an "Unsubscribe" button.

The "Publish:" section includes radio buttons for "Using QOS" (selected), "Ack not required", and "Ack required". Below these is another text input field and a "Publish" button.

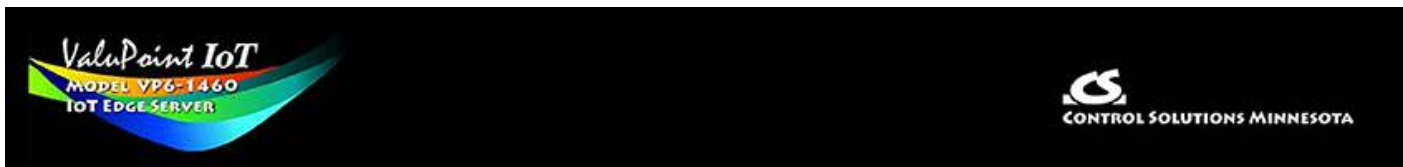
A large white rectangular area is present below the publish options, likely for displaying messages or logs.

At the bottom, there is a section for "Most recent Publish or Subscribe message:" with "Last Pub" and "Last Sub" buttons. Below this is a "Topic:" field containing the text "\$aws/things/aValuPoint/shadow/update".

Below the topic field is a code block showing a JSON message:

```
{ "state":  
  { "desired":  
    { "csiActuator1": 1 }  
  }  
}
```

You should now be able to go to the ValuPoint's Local Registers page and see the value that you published from the MQTT Client.



16. Using Mosquitto MQTT

The ValuPoint supports the use of a generic MQTT broker such as Mosquitto MQTT. This section will review use of Mosquitto. To begin with, you need a Linux PC that can serve as your server, and you will need to install the Mosquitto MQTT software on that Linux server.

16.1 Configuring the Mosquitto Linux Server

To install Mosquitto MQTT if you have not done so already, follow instructions at <http://www.steves-internet-guide.com/mosquitto-broker/>.

The configuration file for Mosquitto is found in `/etc/mosquitto/mosquitto.conf` and the minimum configuration would look like the example below.

A terminal window screenshot showing the command `cat /etc/mosquitto/mosquitto.conf` and its output. The output lists the configuration parameters for Mosquitto, including the pid file, persistence settings, log destination, include directory, and port number.

```
jimhogenson@ubuntu20: /etc/mosquitto
jimhogenson@ubuntu20:/etc/mosquitto$ cat mosquitto.conf
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

pid_file /var/run/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d

port 1883

jimhogenson@ubuntu20:/etc/mosquitto$
jimhogenson@ubuntu20:/etc/mosquitto$
```

Using a bare minimum configuration with no SSL and no username/password, the Thing ID page would look like the following screen shot.

ValuPoint IoT
MODEL VP6-1460
IOT EDGE SERVER

CONTROL SOLUTIONS MINNESOTA

Local Data Modbus IoT Cloud System

Thing Setup Thing Status

Thing Points **Thing ID** Thing Files

Update

Server Host Name

Server Port Disable SSL Disable SSL certificate verify

Thing Name / Client ID

Username

Password

Features Enabled: AWS IoT Core Complex JSON Thingsboard RPC

IoT Engine Status Enabled (See IMPORTANT Note Below)

Subscribe Topics:

Topic 0

Topic 1

Topic 2

Topic 3

Topic 4

Note that 'ubuntu20' as host name has been added to the local DNS server. DNS lookup of 'ubuntu20' returns the IP address of the local Mosquitto MQTT server. The IP address of the local DNS server has also been entered as primary DNS on the Network page in this ValuPoint.

You can also enter the local server's IP address directly as illustrated below if preferred.

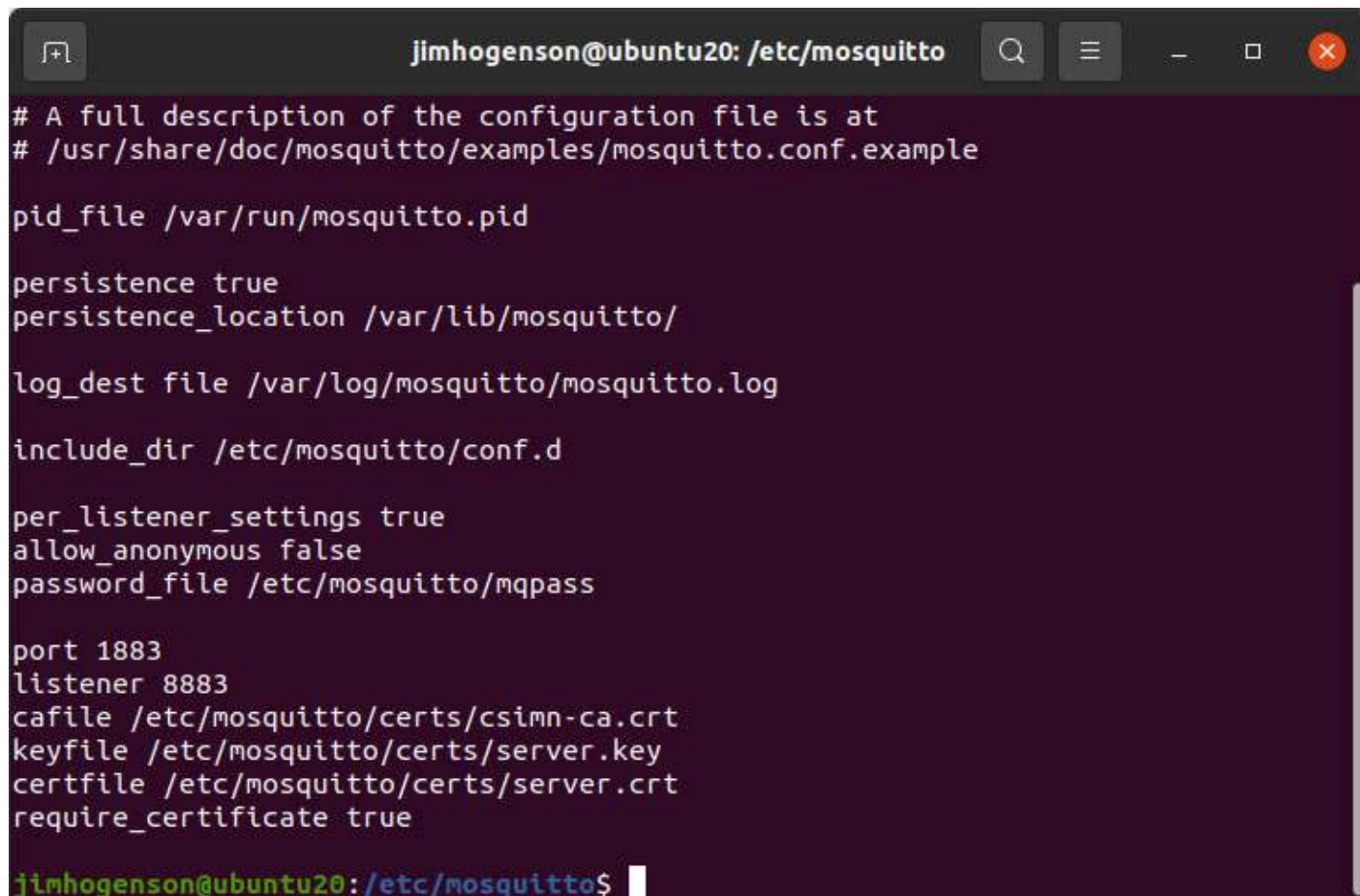
Thing Points **Thing ID** Thing Files

Update

Server Host Name

Server Port Disable SSL Disable SSL certificate verify

Adding both SSL certificates and username/password requirements is illustrated in the mosquitto.conf file pictured below.

A terminal window titled 'jimhogenson@ubuntu20: /etc/mosquitto' showing the contents of a Mosquitto configuration file. The window has a dark background with light-colored text. The configuration includes settings for persistence, logging, include directories, listener settings, and SSL certificates. The prompt at the bottom is 'jimhogenson@ubuntu20:/etc/mosquitto\$'.

```
jimhogenson@ubuntu20: /etc/mosquitto
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

pid_file /var/run/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d

per_listener_settings true
allow_anonymous false
password_file /etc/mosquitto/mqpass

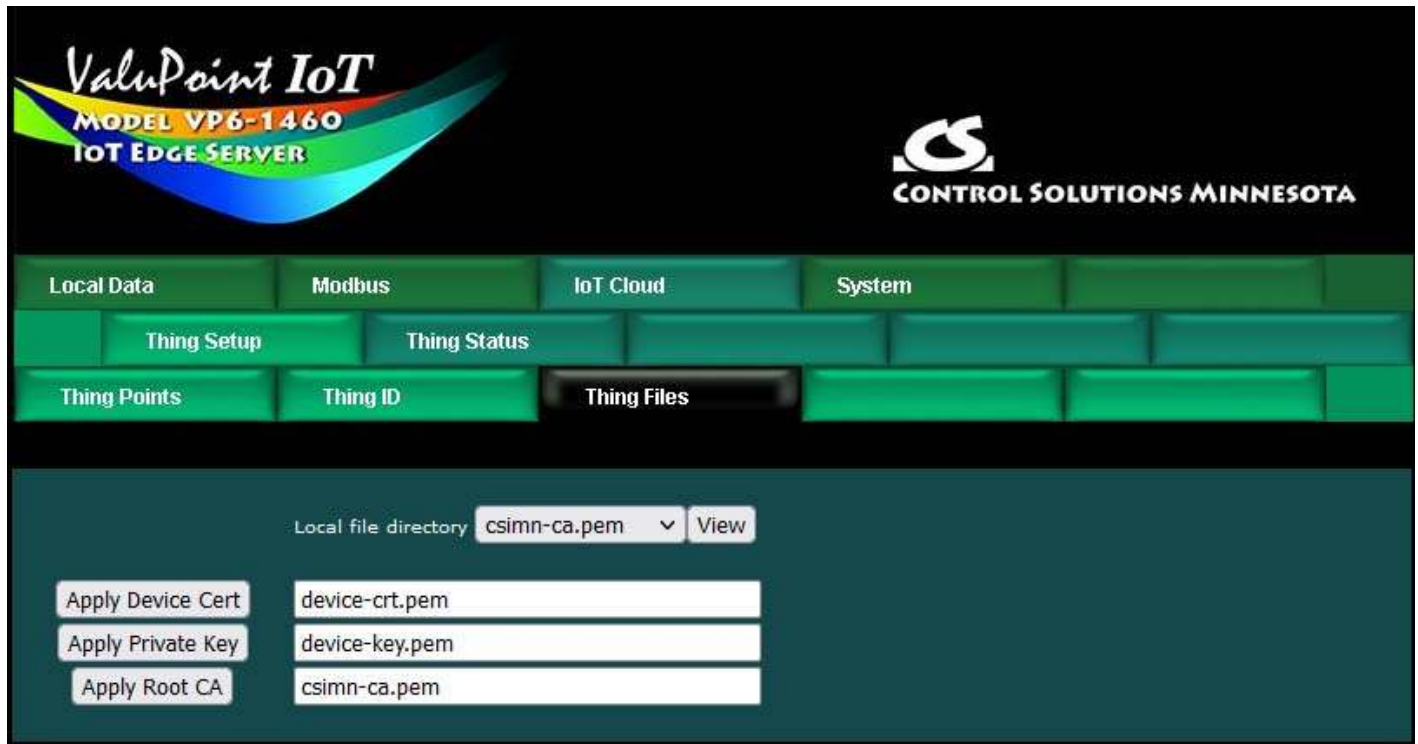
port 1883
listener 8883
cafile /etc/mosquitto/certs/csimn-ca.crt
keyfile /etc/mosquitto/certs/server.key
certfile /etc/mosquitto/certs/server.crt
require_certificate true

jimhogenson@ubuntu20:/etc/mosquitto$
```

Follow instructions for `mosquitto_passwd` (under Documentation at mosquitto.org) for creating the password file and adding usernames to it.

To create your own SSL certificates for both the Mosquitto server and the client (ValuPoint), follow instructions at <http://mosquitto.org/man/mosquitto-tls-7.html> and see also <https://asciinema.org/a/201826>.

Certificates for use with Mosquitto are uploaded and installed in the same manner as for AWS. An example of the Thing Files page is illustrated below.



The Thing ID page when SSL and username/password are configured in Mosquitto would appear as in the screen shot below.

Valupoint IoT
MODEL VP6-1460
IOT EDGE SERVER

CONTROL SOLUTIONS MINNESOTA

Local Data | Modbus | IoT Cloud | System

Thing Setup | Thing Status

Thing Points | **Thing ID** | Thing Files

Update

Server Host Name:

Server Port: Disable SSL Disable SSL certificate verify

Thing Name / Client ID:

Username:

Password:

Features Enabled: AWS IoT Core Complex JSON Thingsboard RPC

IoT Engine Status: Enabled (See IMPORTANT Note Below)

Subscribe Topics:

Topic 0:

Topic 1:

Topic 2:

Topic 3:

Topic 4:

Upon successful connection, you should see the "success" indication as pictured below.

ValuPoint IoT
MODEL VP6-1460
IOT EDGE SERVER

CONTROL SOLUTIONS MINNESOTA

Local Data Modbus IoT Cloud System

Thing Setup Thing Status

Object Info Connection Test

Clear Refresh

Connection Status

Failed Connection Count

Publish Message Count

Publish Error Count

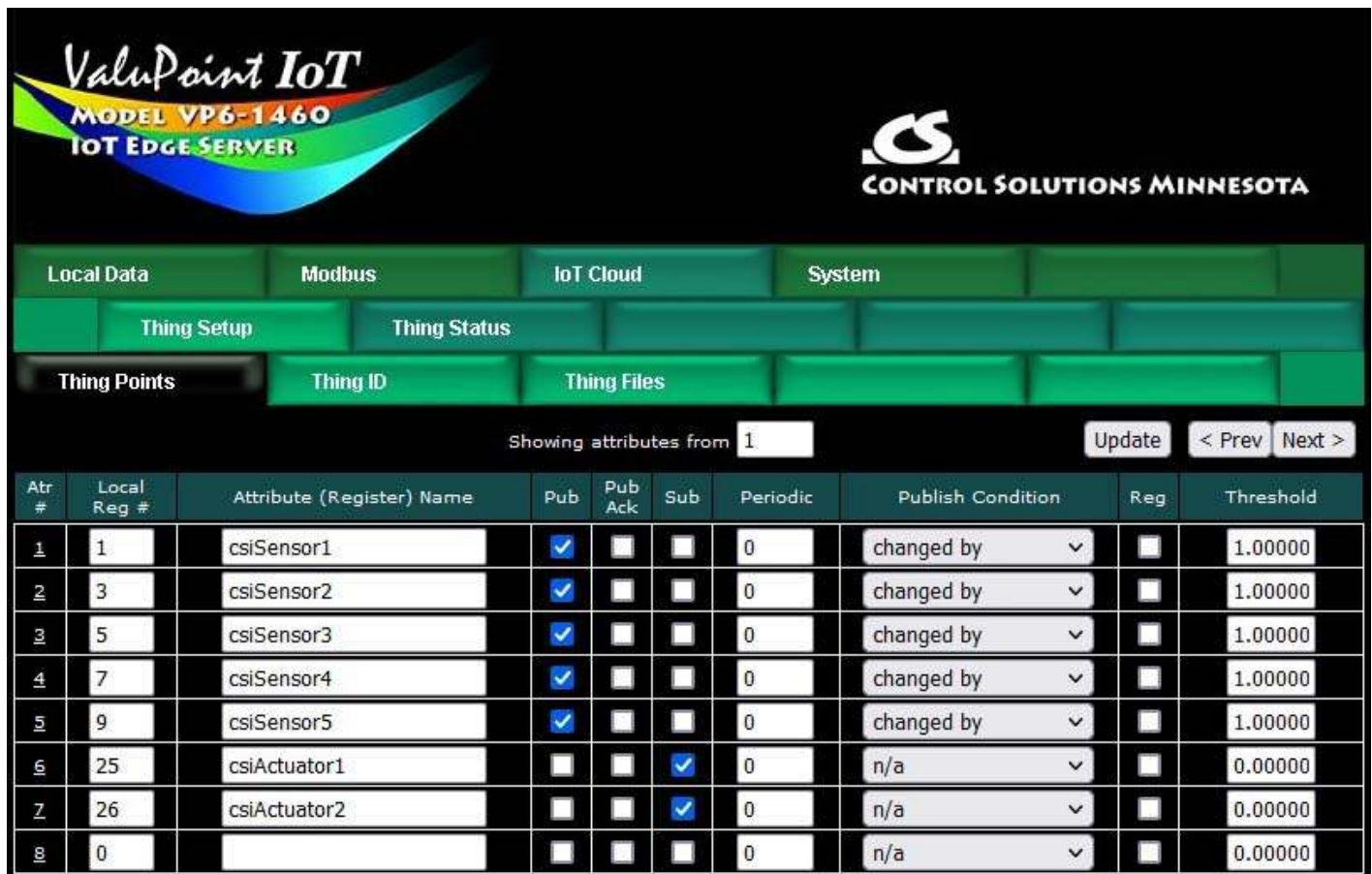
Subscribe Message Count

Subscribe Error Count

Connection Info Connecting to server at ubuntu20:8883.
Connected securely!

16.2 Publishing Thing Points to Mosquitto

The same set of Thing Points, along with the same publish and subscribe rules, as used for AWS will work the same with Mosquitto MQTT or any other MQTT broker. Refer to section 13 for Thing Point configuration.



Valupoint IoT
MODEL VP6-1460
IoT EDGE SERVER

CONTROL SOLUTIONS MINNESOTA

Local Data Modbus IoT Cloud System

Thing Setup Thing Status

Thing Points Thing ID Thing Files

Showing attributes from 1 Update < Prev Next >

Atr #	Local Reg #	Attribute (Register) Name	Pub	Pub Ack	Sub	Periodic	Publish Condition	Reg	Threshold
1	1	csiSensor1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by	<input type="checkbox"/>	1.00000
2	3	csiSensor2	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by	<input type="checkbox"/>	1.00000
3	5	csiSensor3	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by	<input type="checkbox"/>	1.00000
4	7	csiSensor4	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by	<input type="checkbox"/>	1.00000
5	9	csiSensor5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by	<input type="checkbox"/>	1.00000
6	25	csiActuator1	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	0	n/a	<input type="checkbox"/>	0.00000
7	26	csiActuator2	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	0	n/a	<input type="checkbox"/>	0.00000
8	0		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	n/a	<input type="checkbox"/>	0.00000

The following screen shot shows using the `mosquitto_sub` utility to subscribe to the default topic for testing our IoT device's publish to that topic. The `mosquitto_sub` is among the utilities installed when you install Mosquitto on your Linux server. Refer to mosquitto.org Documentation for further instructions on using `mosquitto_sub`.

```
jimhogenson@ubuntu20: ~  
/object/update { "state": { "desired": { "csiActuator1": 777 } } }  
/object/update { "state": { "desired": { "csiActuator1": 777 } } }  
^Cjimhogenson@ubuntu20:~$ mosquitto_sub -h localhost -p 1883 -t '/object/update'  
-v --username ubuntu --pw ub20pass  
/object/update { "state": { "desired": { "csiActuator1": 777 } } }  
/object/update { "state": { "desired": { "csiActuator1": 777 } } }  
^Cjimhogenson@ubuntu20:~$ mosquitto_sub -h localhost -p 1883 -t '/default' -v --  
username ubuntu --pw ub20pass  
/default { "state": { "reported": { "csiSensor1": 0.00, "csiSensor2": 11.000000,  
"csiSensor3": 0.00, "csiSensor4": 0.00, "csiSensor5": 0.00, "csiSensor10": "IoT  
sensor state 1", "LocalTime": "2022-10-13T09:17:53-05:00" } } }  
/default { "state": { "reported": { "csiSensor1": 0.00, "csiSensor2": 6.000000,  
"csiSensor3": 0.00, "csiSensor4": 0.00, "csiSensor5": 0.00, "csiSensor10": "IoT  
sensor state 1", "LocalTime": "2022-10-17T09:33:10-05:00" } } }  
/default { "state": { "reported": { "csiSensor1": 0.00, "csiSensor2": 2.000000,  
"csiSensor3": 0.00, "csiSensor4": 0.00, "csiSensor5": 0.00, "csiSensor10": "IoT  
sensor state 1", "LocalTime": "2022-10-17T09:41:55-05:00" } } }  
/default { "state": { "reported": { "csiSensor1": 0.00, "csiSensor2": 9.000000,  
"csiSensor3": 0.00, "csiSensor4": 0.00, "csiSensor5": 0.00, "csiSensor10": "IoT  
sensor state 1", "LocalTime": "2022-10-17T09:42:03-05:00" } } }  
/default { "state": { "reported": { "csiSensor1": 0.00, "csiSensor2": 0.00, "csi  
Sensor3": 0.00, "csiSensor4": 0.00, "csiSensor5": 0.00, "csiSensor10": "IoT sens  
or state 1", "LocalTime": "2022-10-17T09:42:17-05:00" } } }  
|
```

The following screen shot shows an example of publishing from the test client to our IoT device using the `mosquitto_pub` utility. This example was created prior to adding username/password to this instance of the broker.

```
jimhogenson@ubuntu20: ~  
jimhogenson@ubuntu20:~$ mosquitto_pub -h localhost -p 1883 -t '/object/update' -  
m '{ "state": { "desired": { "csiActuator1": 234 } } }'  
jimhogenson@ubuntu20:~$ |
```

The JSON expected by AWS IoT Core is a complex object structure. You have the option of keeping this complex structure, or using "simple" JSON. Some applications may require just simple JSON. To switch to simple JSON, just un-select Complex JSON on the Thing ID page as illustrated below.

Valupoint IoT
MODEL VP6-1460
IOT EDGE SERVER

CONTROL SOLUTIONS MINNESOTA

Local Data Modbus IoT Cloud System

Thing Setup Thing Status

Thing Points **Thing ID** Thing Files

Update

Server Host Name

Server Port Disable SSL Disable SSL certificate verify

Thing Name / Client ID

Username

Password

Features Enabled: AWS IoT Core Complex JSON Thingsboard RPC

IoT Engine Status Enabled (See IMPORTANT Note Below)

Subscribe Topics:

Topic 0

Topic 1

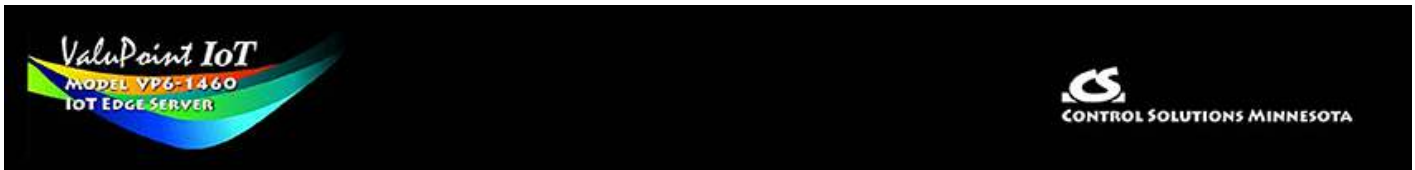
Topic 2

Topic 3

Topic 4

The screen shot below shows publishing "simple" JSON to the broker which in turn will forward this message to the MQ device. Compare this to the mosquitto_pub example above.


```
jimhogenson@ubuntu20: ~  
jimhogenson@ubuntu20:~$ mosquitto_pub -h localhost -p 1883 -t '/object/update' -  
m '{ "csiActuator1": 55 }'  
Connection error: Connection Refused: not authorised.  
Error: The connection was refused.  
jimhogenson@ubuntu20:~$ mosquitto_pub -h localhost -p 1883 --username ubuntu --p  
w ub20pass -t '/object/update' -m '{ "csiActuator1": 55 }'  
jimhogenson@ubuntu20:~$ mosquitto_pub -h localhost -p 1883 --username ubuntu --p  
w ub20pass -t '/object/update' -m '{ "csiActuator1": 55, "csiActuator2":12, "csi  
Actuator3":112 }'  
jimhogenson@ubuntu20:~$ mosquitto_pub -h localhost -p 1883 --username ubuntu --p  
w ub20pass -t '/object/update' -m '{ "csiActuator1": 66, "csiActuator2":12, "csi  
Actuator3":112 }'  
jimhogenson@ubuntu20:~$
```



17. Using Thingsboard.io MQTT

17.1 Introduction to Thingsboard

Thingsboard provides a number of capabilities including interactive real time dashboards. Here is a screen shot of the demo dashboard that will be built as you follow through this section of this user guide.



We refer to "MQTT device" throughout the discussion that follows because the discussion is generic to any of the Control Solutions products that have MQTT capability including the ValuPoint IoT Edge Servers and the Babel Buster IoT gateways like BB3-6101-MQ.

Start by signing up for a Thingsboard account (no cost) here: <https://demo.thingsboard.io/signup>

Once you have an account, you can log in here: <https://demo.thingsboard.io/login>

The home page contains a number of links to tutorials and documentation.

17.2 Adding a Device

To add a new device, select the Devices page (under Entities) from the list on the left. Then click "+" in upper right corner and "Add new device".

Add new device ? X

1 Device details **2** Credentials Optional

Name*
VP6-1460

Label
Test Device VP6-1460

Device profile*
default X ✎

Is gateway

Assign to customer

Description

Next: Credentials

Cancel Add

Give the device a name and label, and select the default profile. Click "Next: Credentials" in the lower right.

Add new device

? ×

✓ Device details 2 Credentials Optional

Credentials type

Access token X.509 **MQTT Basic**

Client ID
vp6-1460 📄

User Name*
testDevice1460 📄

Password
[blurred] 👁 📄

Back

Cancel Add

On the Credentials page, select MQTT Basic. You can convert to more secure methods later if desired. Provide a client ID, and this must be unique across all of your devices as this ID is how Thingsboard will identify where data is coming from. Provide a username and password for this device. This username and password will be used by the device, not for logging into your Thingsboard account. Do not use your account login credentials here.

Click the Add button. This now adds the new device to your account. Thingsboard will automatically take you to the following screen. Skip this. You will not be using your PC to connect to Thingsboard. You will be using your MQTT device and it already has all

the software it needs.

Device created. Let's check connectivity! ✕

Use the following instructions for sending telemetry on behalf of the device using shell

Windows
 MacOS
 Linux
 Docker

Install necessary client tools Skip This.

Use the instructions to download, install, setup and run mosquitto_pub Documentation

Execute the following command

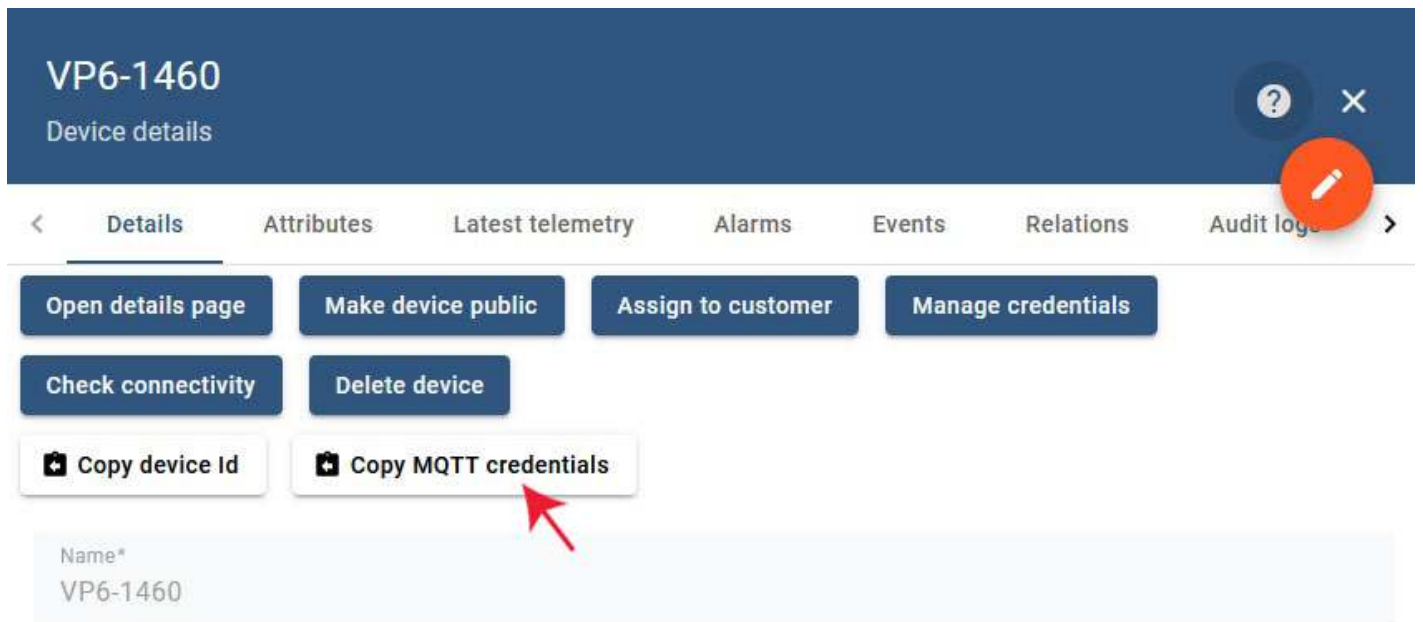
```
mosquitto_pub -d -q 1 -h demo.thingsboard.io -p 1883 -t v1/devices/
```

The newly added device will now show up on your device list with the newest device typically at the top. Thingsboard automatically creates some demo devices for you when you first create your account. You can ignore those.

Devices Device Filter

<input type="checkbox"/>	Created time ↓	Name	Device profile	Label	State	Customer
<input type="checkbox"/>	2024-03-12 09:20:43	VP6-1460	default	Test Device VP6-1460	Inactive	
<input type="checkbox"/>	2022-11-07 08:57:04	MQ73	default	Test Device 2	Inactive	
<input type="checkbox"/>	2022-11-04 10:00:48	MQ61	default	Test Device	Inactive	

The State will be either Inactive or Active. Whatever it is, click on the state to see device details.



VP6-1460
Device details

< Details Attributes Latest telemetry Alarms Events Relations Audit log >

Open details page Make device public Assign to customer Manage credentials

Check connectivity Delete device

Copy device Id Copy MQTT credentials

Name*
VP6-1460

Click on Copy MQTT credentials, and paste the result into a temporary text document created with Notepad or equivalent. You will use these credentials shortly. You can skip this copying if you had already written down all of the credentials when you first entered them above, but if you copy, you can also paste into the web page for the device.

The credentials string you will get by the Copy is illustrated below.



```
*Untitled - Notepad
File Edit Format View Help
{clientId:"vp6-1460",userName:"testDevice1460",password:"..."}
```

The screenshot shows the configuration page for a device in the Thingsboard.io interface. The page is titled "ValuPoint IoT MODEL VP6-1460 IOT EDGE SERVER" and is managed by "CONTROL SOLUTIONS MINNESOTA". The navigation menu includes "Local Data", "Modbus", "IoT Cloud", "System", "Thing Setup", "Thing Status", "Thing Points", "Thing ID", and "Thing Files". The "Thing ID" page is active, showing the following configuration:

- Server Host Name: demo.thingsboard.io
- Server Port: 1883, with checkboxes for "Disable SSL" (checked) and "Disable SSL certificate verify" (unchecked).
- Thing Name / Client ID: vp6-1460
- Username: testDevice1460
- Password: [Redacted]
- Features Enabled: AWS IoT Core, Complex JSON, Thingsboard RPC
- IoT Engine Status: Enabled (See IMPORTANT Note Below)
- Subscribe Topics:
 - Topic 0: v1/devices/me/rpc/request/+
 - Topic 1: [Empty]
 - Topic 2: [Empty]
 - Topic 3: [Empty]
 - Topic 4: [Empty]

An "Update" button is located in the top right corner of the configuration area.

Log into your VP6-1460 and go to the Thing ID page. For MQTT Basic credentials, the following settings will always be the same:

- * Server Host Name: demo.thingsboard.io
- * Server Port: 1883, Disable SSL
- * Unselect AWS IoT Core and Complex JSON. Select Thingsboard RPC

Enter your device's client ID, username, and password.

Leave the default topic illustrated above as is. Click Update. If you will not be doing any additional configuration right now, go to the File Manager page and save your XML file. Otherwise continue with setup and save the XML later. Just remember to save the XML file at some point as this is where your Thing ID information is stored.

Once your "Thing" is online, and it publishes for the first time, you can verify the publish on the Object Info page.

ValuPoint IoT
MODEL VP6-1460
IOT EDGE SERVER

CONTROL SOLUTIONS MINNESOTA

Local Data Modbus IoT Cloud System

Thing Setup Thing Status

Object Info Connection Test

Showing attributes from 1 Refresh < Prev Next >

Atr #	Attribute Name	Last Exchange Timestamp	Last Exchanged Value (or Status)
1	csiSensor1	Pub@ 2024-03-12 09:33:37	3.191312
2	csiSensor2	Pub@ 2024-03-12 09:33:37	0.002785
3	csiSensor3	Pub@ 2024-03-12 09:33:37	0.000000
4	csiSensor4	Pub@ 2024-03-12 09:33:37	0.005569
5	csiSensor5	Pub@ 2024-03-12 09:33:37	0.000000
6	csiActuator1		
7	csiActuator2		
8			

As soon as Thingsboard sees data from your device, the state will change to Active as illustrated below.

Devices

<input type="checkbox"/>	Created time ↓	Name	Device profile	Label	State	Customer
<input type="checkbox"/>	2024-03-12 09:20:43	VP6-1460	default	Test Device VP6-1460	Active	


Click on "Active" and you will see your device details. Select the Telemetry tab and you will see the data most recently sent.

VP6-1460

Device details

?







✕



< Details Attributes **Latest telemetry** Alarms Events Relations Audit logs >

Telemetry

+ 🔍

<input type="checkbox"/>	Last update time	Key ↑	Value	
<input type="checkbox"/>	2024-03-12 10:03:20	csiSensor1	2.534113	
<input type="checkbox"/>	2024-03-12 10:03:20	csiSensor2	0.005569	
<input type="checkbox"/>	2024-03-12 10:03:20	csiSensor3	0.0	
<input type="checkbox"/>	2024-03-12 10:03:20	csiSensor4	0.005569	
<input type="checkbox"/>	2024-03-12 10:03:20	csiSensor5	0.002785	
<input type="checkbox"/>	2024-03-12 10:03:20	LocalTime	2024-03-12T10:03:20-05:00	

Configuring the Thing Points to publish to Thingsboard follows all the same guidelines as for publishing to AWS. The only difference is the MQTT Topic. To publish sensor data that you wish to display on a dashboard, publish to the topic "v1/devices/me/telemetry".

Local Data Modbus IoT Cloud System

Thing Setup Thing Status

Thing Points Thing ID Thing Files

Attribute # 1 Update < Prev Next >

Associate local register # 1 named csiSensor1 with this IoT attribute.

Publish: Using QOS Ack not required Ack required Publish as Reported Desired

MQTT Topic: Default Other v1/devices/me/telemetry

Publish if register value is changed by this value: 1.000000 this local register: 0

Qualified by this hysteresis value: 0.000000 this minimum On Time: 0:00:00 this minimum Off Time: 0:00:00

Publish at least every 0 minutes. Publish no more than every 0 minutes.

Follow above rule only if local register 0 is set to a value of 0

Publish message on true:

Publish message on false:

Publish as part of dataset number: 1 Include timestamp

Subscribe: To topic index: 0 v1/devices/me/rpc/request/+

Apply this default value: 0.000000 after 0 minutes without any update from the cloud.

Attributes Enabled: 8 Make Template Force Publish Insert Delete

Configuring the Thing Points to subscribe to Thingsboard follows all the same guidelines as for subscribing to AWS. The only difference is the MQTT Topic. To subscribe to a topic that lets you control points from the dashboard, subscribe to the topic "v1/devices/me/rpc/request/+"

The screenshot shows the configuration page for a Thing Point in Thingsboard.io. The interface is organized into several sections:

- Navigation:** Local Data, Modbus, IoT Cloud, System, Thing Setup, Thing Status, Thing Points (selected), Thing ID, Thing Files.
- Attribute #:** 6. Buttons: Update, < Prev, Next >
- Associate:** local register # 25 named CSIActuator1 with this IoT attribute.
- Publish:**
 - Using QOS: Using QOS, Ack not required, Ack required
 - Publish as: Reported, Desired
 - MQTT Topic: Default, Other: v1/devices/me/rpc/request/+
 - Publish if register value is: n/a (dropdown), this value: 0.000000, this local register: 0
 - Qualified by this hysteresis value: 0.000000, this minimum On Time: 0:00:00, this minimum Off Time: 0:00:00
 - Publish at least every 0 minutes. Publish no more than every 0 minutes.
 - Follow above rule only if local register 0 is set to a value of 0
 - Publish message on true: [text input]
 - Publish message on false: [text input]
 - Publish as part of dataset number: 0, Include timestamp
- Subscribe:** To topic index: 0 v1/devices/me/rpc/request/+
- Apply this default value: 0.000000 after 0 minutes without any update from the cloud.
- # Attributes Enabled:** 8. Buttons: Make Template, Force Publish, Insert, Delete

For our dashboard demo, we have created the list of points illustrated below. The lower numbered registers are the physical I/O points. The higher numbered registers are mapped to Modbus queries of a remote Modbus device.

Local Data		Modbus		IoT Cloud		System						
Thing Setup			Thing Status									
Thing Points			Thing ID			Thing Files						
Showing attributes from 1										Update	< Prev	Next >
Atr #	Local Reg #	Attribute (Register) Name	Pub	Pub Ack	Sub	Periodic	Publish Condition	Reg	Threshold			
<u>1</u>	1	csiSensor1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by ▾	<input type="checkbox"/>	1.00000			
<u>2</u>	3	csiSensor2	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by ▾	<input type="checkbox"/>	1.00000			
<u>3</u>	5	csiSensor3	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by ▾	<input type="checkbox"/>	1.00000			
<u>4</u>	7	csiSensor4	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by ▾	<input type="checkbox"/>	1.00000			
<u>5</u>	9	csiSensor5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by ▾	<input type="checkbox"/>	1.00000			
<u>6</u>	25	csiActuator1	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	0	n/a ▾	<input type="checkbox"/>	0.00000			
<u>7</u>	26	csiActuator2	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	0	n/a ▾	<input type="checkbox"/>	0.00000			
<u>8</u>	27	csiSensor6	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by ▾	<input type="checkbox"/>	1.00000			
<u>9</u>	28	csiSensor7	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by ▾	<input type="checkbox"/>	1.00000			
<u>10</u>	29	csiSensor8	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by ▾	<input type="checkbox"/>	1.00000			
<u>11</u>	30	csiSensor9	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	changed by ▾	<input type="checkbox"/>	1.00000			
<u>12</u>	32	csiActuator3	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	0	n/a ▾	<input type="checkbox"/>	0.00000			
<u>13</u>	33	csiActuator4	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	0	n/a ▾	<input type="checkbox"/>	0.00000			
<u>14</u>	0		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0	n/a ▾	<input type="checkbox"/>	0.00000			

17.3 Creating a Dashboard

Click on Dashboard near the top of the list of menu items on the left in your Thingsboard account. You will find a number of example dashboard provided by Thingsboard. Feel free to look around.

To create your own dashboard, click the "+" icon in the upper right corner of the Dashboard page, and select Create New Dashboard. Give the dashboard a name (and description, optional). You don't need to do anything else at this point other than click Add.

Add dashboard

?

X

Title*
Dashboard-1460

Description
Demo dashboard for VP6-1460

Assigned customers

Mobile application settings

Hide dashboard in mobile application

Dashboard order in mobile application

Dashboard image

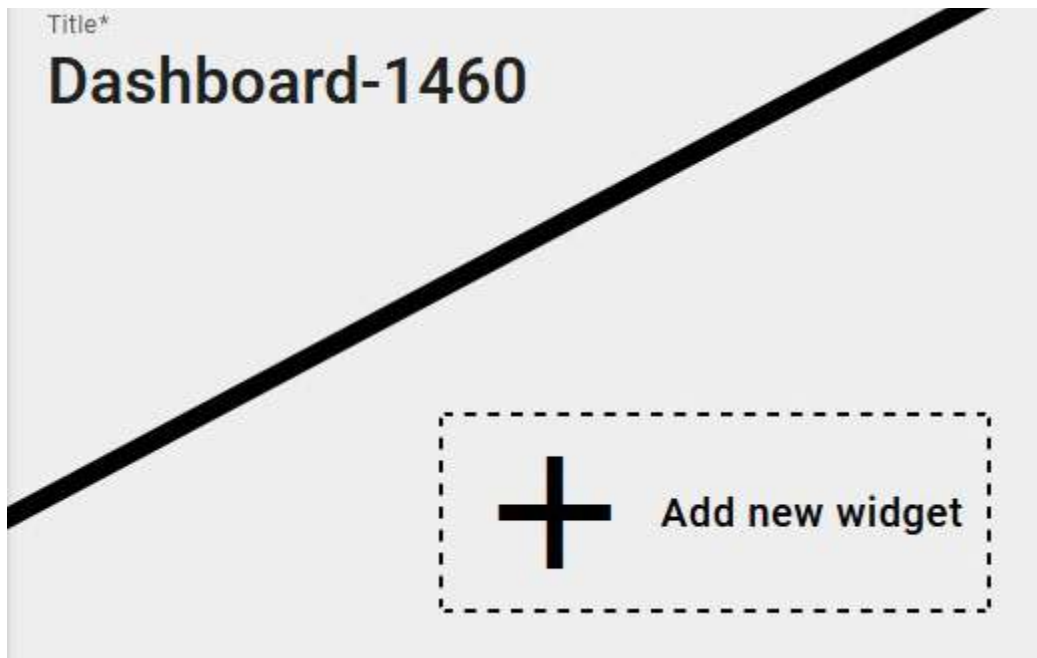
No image selected

Browse from gallery

Set link

Cancel Add

As soon as you click Add, you will see a mostly blank page with "Add new widget" in the middle. Click that add button.



As soon as you click the Add widget, you will see a screen of categories of widgets available. There are many widgets to pick from.

Title*
Dashboard-1460

Select widgets bundle

🔍 Import widget

Charts sys i

Cards sys i

Alarm widgets sys i

Count widgets sys i

Maps sys i

Analogue gauges sys i

Status indicators sys i

Industrial widgets sys i

Indoor Environment sys i

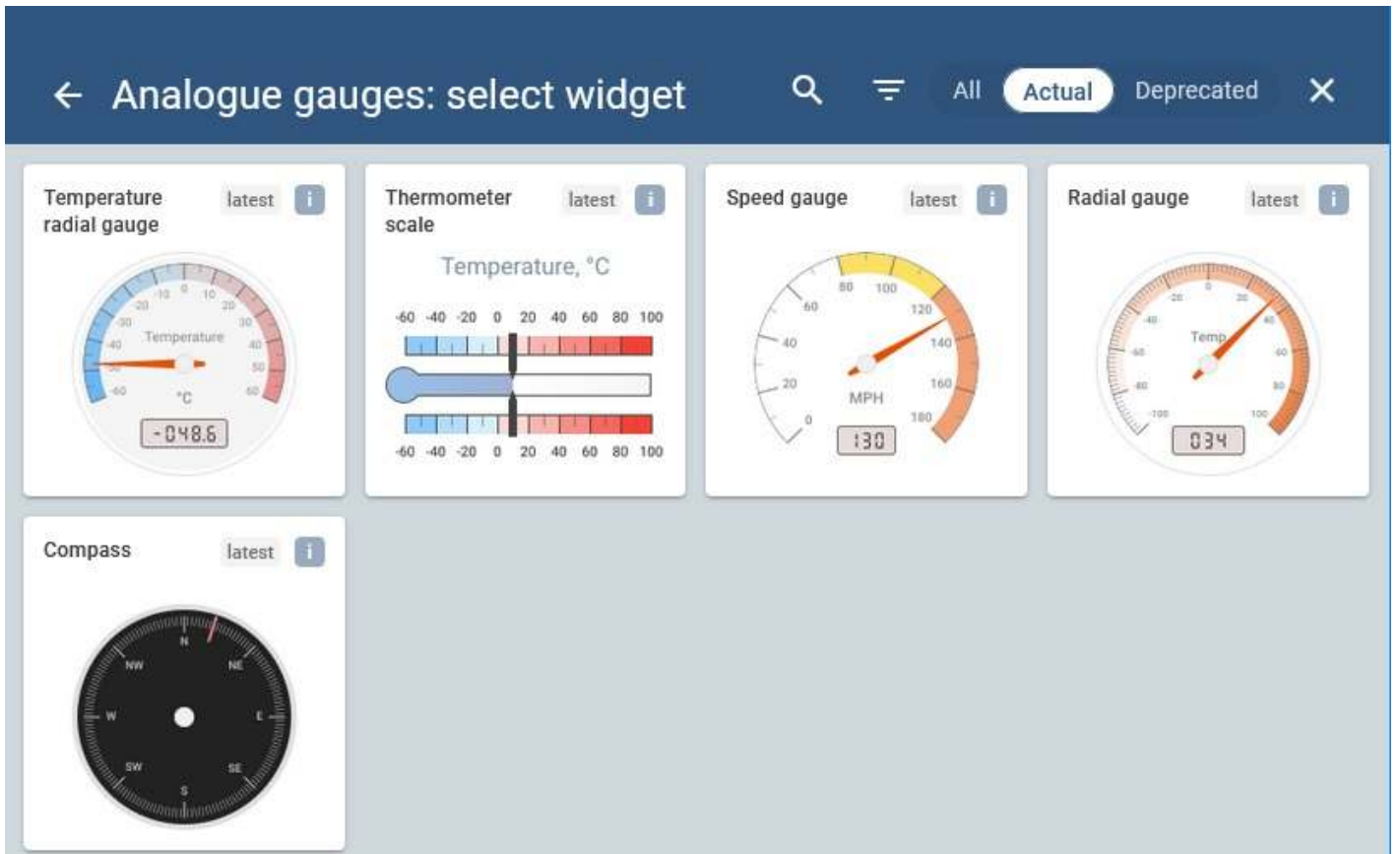
Liquid level sys i

Digital gauges sys i

Air quality sys i

17.4 Adding a Sensor Widget

Click Add widget. For our demo here, we will select Analogue gauges. After clicking that icon (illustrated above), the screen below will appear. Select the gauge type you desire.



We are selecting the first gauge, Temperature, for this demo.

Add widget: Temperature radial gauge

Basic Advanced ? X

Datasource Device Entity alias

Device*
VP6-1460 X

Data key*
csiSensor6 X

Gauge appearance

Units title A [Color]

Units A [Color]

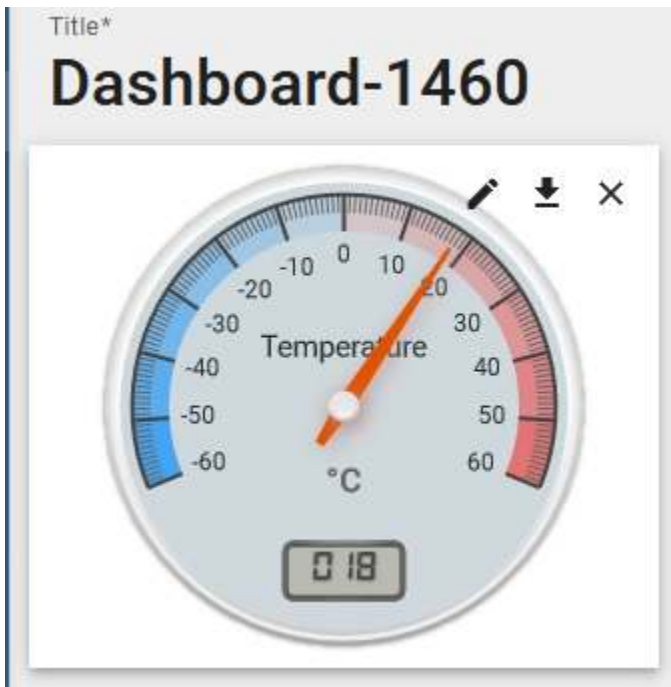
Value decimals A [Color]

Ticks min max A [Color]


Cancel Preview **Add**

Your device name should appear in the Device list. Select it. Delete any default Data key that appears and then the list will be populated with your recently published data points. Select the name associated with the Thing Point which you configured on the Thing Points page in the VP6-1460. This tells Thingsboard which data to associate with this gauge. The first two items illustrated above are the only required items. The rest of the parameters are optional formatting that you can play with later.


Click Add, and you should see the gauge illustrated below on your dashboard.



Our point csiSensor6 is being displayed on the gauge, and it is getting its data from a remote Modbus device.



MODEL VP6-1460
IOT EDGE SERVER



CONTROL SOLUTIONS MINNESOTA

Local Data
Modbus
IoT Cloud
System

RTU Setup
RTU Data
TCP Setup
TCP Data

Devices
Client Read Map
Client Write Map

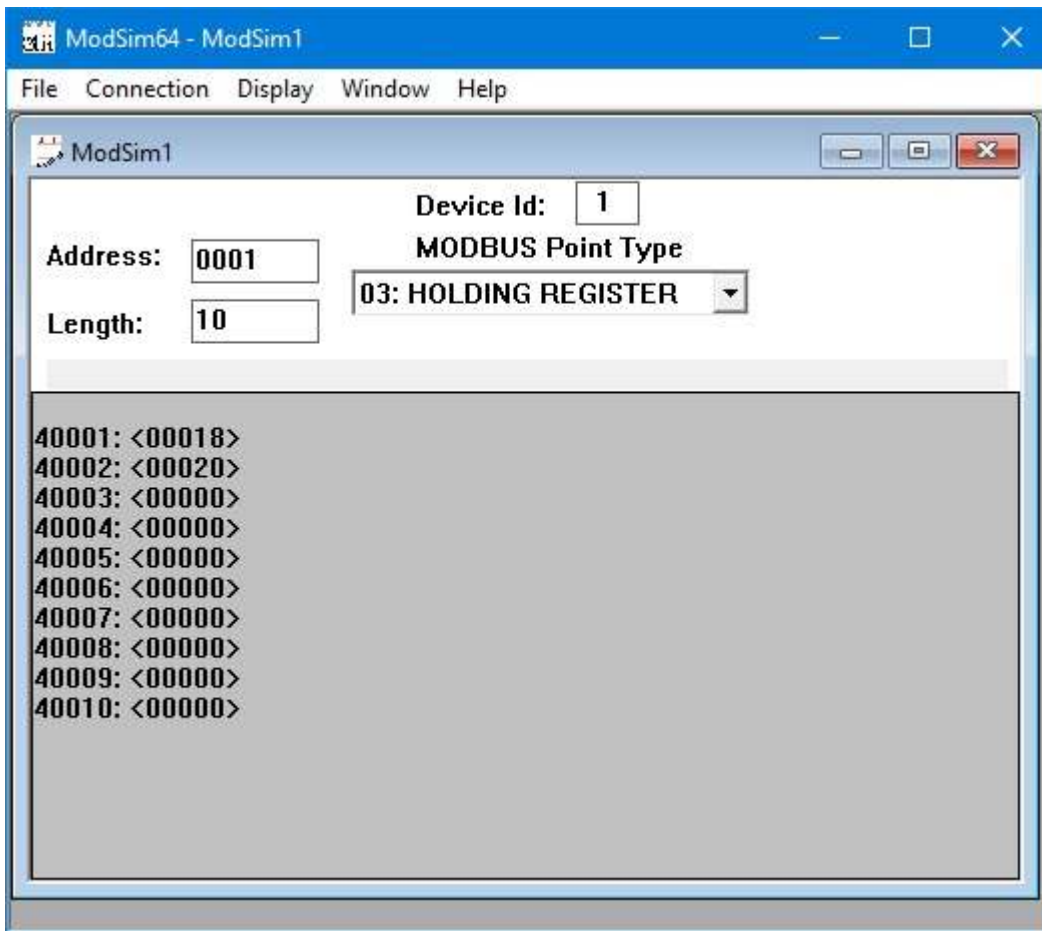
Showing 1 to 6 of 6
 Update
< Prev
Next >

Map #	Remote Type	Remote Register Format	Remote Register #	Remote Device	Local Register #	Name
1	Holding Register ▾	UINT-16 ▾	1	ModSim ▾	27	csiSensor6
2	Holding Register ▾	UINT-16 ▾	2	ModSim ▾	28	csiSensor7
3	Holding Register ▾	UINT-16 ▾	3	ModSim ▾	29	csiSensor8
4	Holding Register ▾	UINT-16 ▾	4	ModSim ▾	30	csiSensor9
5	Holding Register ▾	UINT-16 ▾	5	ModSim ▾	31	csiSensor10
6	None ▾	None ▾	0	None ▾	0	

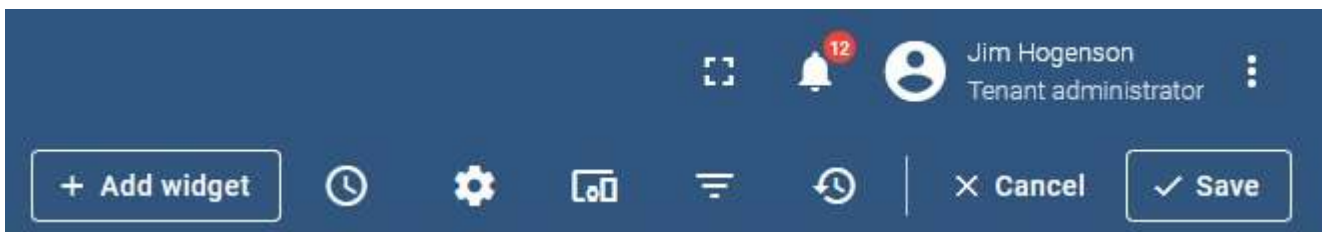
For test purposes, our remote Modbus device is ModSim. If we go into ModSim and change the data for its register #1, that number will show up on the gauge.

You will note that the gauge defaults to displaying both positive and negative numbers but we are using an unsigned integer for Modbus data. That means the gauge will never display negative temperatures. You can either alter the gauge, or alter the

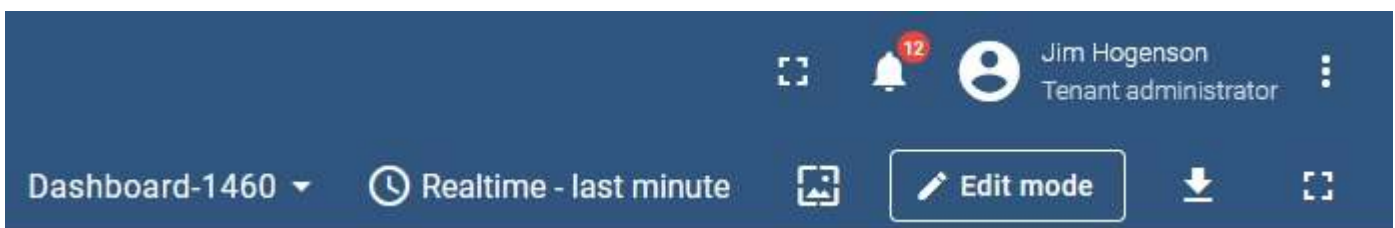
configuration in the VP6-1460 to correct this situation.



When you are done making changes to the dashboard, click the Save button.



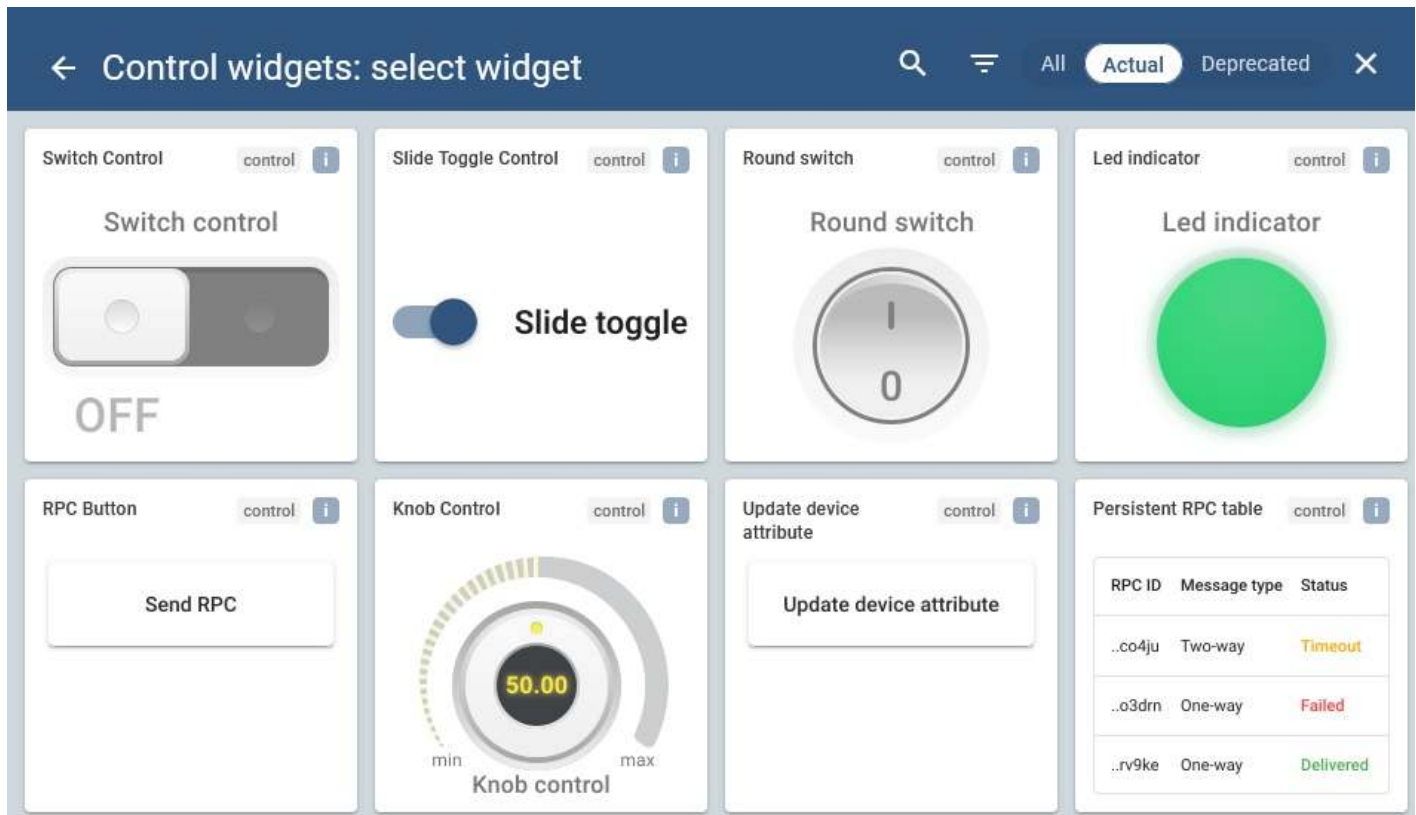
To resume modifying your dashboard, click the Edit mode button, which may show up as just the pencil icon.



17.5 Adding an Actuator Widget

Adding a widget to control a parameter in your MQTT device begins by clicking the Add widget button in your dashboard (in edit mode). Select the Control widgets category.

Then pick which widget you want.



For our first widget, we will select the Know Control. Select your device from the device list.

Add widget: Knob Control

Target device

Device*
|

- Test Device A3
- Test Device B1
- Test Device C1
- VP6-1460
- VP6-1470

Cancel Preview Add

Do not click Add just yet.

Add widget: Knob Control

Target device

Device*
VP6-1460

Click on the Appearance tab. There are a couple of very important things you need to do on the Appearance tab. Enter "get_XXX" for RPC get value method and "set_XXX"

for RPC set value method where XXX is the attribute name assigned in your MQTT device on the Thing Points page.

In our example illustrated above and below, our Thing Point name is csiActuator3. Thus we must enter `get_csiActuator3` and `set_csiActuator3` for the RPC methods. These RPC method names will be processed by our MQTT device (VP6-1460) when Thingsboard wants to exchange data.

If you experience timeouts, increase the timeout value on this tab.

Add widget: Knob Control

Appearance

Common settings

Knob title
Knob control

Value settings

Initial value
50

Minimum value*
0

Maximum value*
100

RPC get value method*
get_csiActuator3

RPC set value method*
set_csiActuator3

RPC settings

RPC request timeout (ms)*
5000

Cancel Preview **Add**

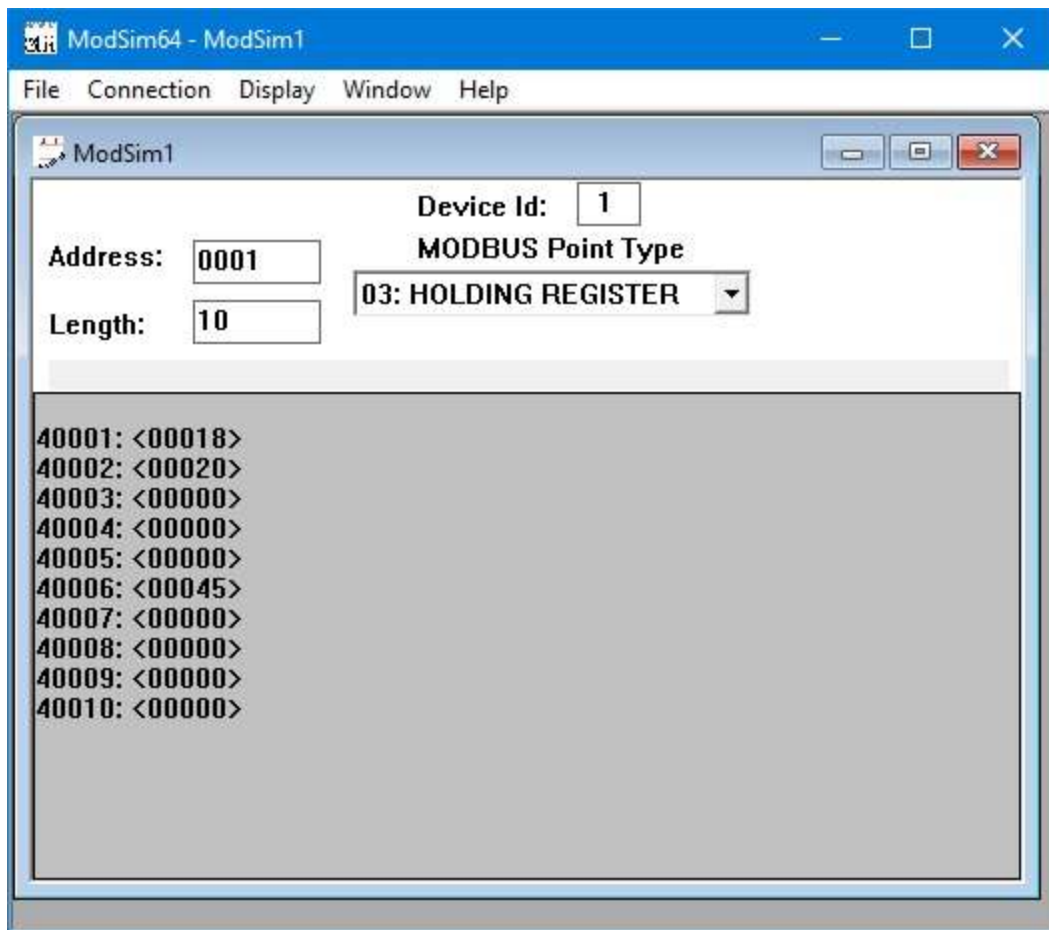
Click Add. The control will now appear on our dashboard.



Our csiActuator3 is mapped to a remote Modbus device, namely ModSim as in our previous demo.

RTU Setup		RTU Data		TCP Setup		TCP Data	
Devices		Client Read Map		Client Write Map			
Showing 1 to 6 of 6				Update		< Prev Next >	
Map #	Local Register #	Remote Type	Remote Register Format	Remote Register #	Remote Device	Name	
1	32	Holding Register	UINT-16	6	ModSim	csiActuator3	
2	33	Holding Register	UINT-16	7	ModSim	csiActuator4	
3	34	Holding Register	UINT-16	8	ModSim	csiActuator5	
4	35	Holding Register	UINT-16	9	ModSim	csiActuator6	
5	36	Holding Register	UINT-16	10	ModSim	csiActuator7	
6	0	None	None	0	None		

Any time we change the dial setting on the dashboard screen, the value displayed in ModSim will correspond very shortly.



The process for adding a switch begins the same as for a knob. Create a new alias, and select our MQTT device from the list.

Add alias ✕

Alias name*
VP6-1460-Switch1

Resolve as multiple entities

Filter type*
Single entity

Type*
Device

Device*
VP6-1460

Cancel Add

The "Initial value" will default to unselected. Select it by checking the box. Select "Call RPC get value method" from the method list if not already the default, and enter the RPC method as described above. We want to remotely switch our relay on and off, so we have entered `get_csiActuator1`. Leave the Parse value function as is, as illustrated below.

Switch Control

Switch Control

Appearance

Widget card Actions Preview Decline Apply

Data settings

"No data to display" alternative message Set message

Common settings

Switch title
Switch control

Show on/off labels

Value settings

Initial value

Retrieve on/off value settings

Retrieve value using method
Call RPC get value method

RPC get value method*
get_csiActuator1

Parse value function: f(data) Tidy ?

```
1 return data ? true : false;
```

Scroll down the the Update value settings portion of the screen. Enter the set_XXX function for the RPC set value method as previously discussed. In this case we use set_csiActuator1.

Add widget: Switch Control

Appearance

Update value settings

RPC set value method*
set_csiActuator1

Convert value function: f(value)

```
1 return value;
```

RPC settings

RPC request timeout (ms)
5000

Persistent RPC settings

Cancel Preview Add

Click Add. We now have a switch on our dashboard.



The Thing Points configuration that makes this happen on the MQTT device side is illustrated (abbreviated form) below. Nothing about publish is needed here, just subscribe.

When the Switch control is clicked on the screen, the corresponding relay switches on and off.

The screenshot shows the configuration page for a "Thing Point". At the top, there are tabs for "Thing ID", "Thing Files", and "Thing ID". Below the tabs, there is an "Attribute # 6" field and "Update", "< Prev", and "Next >" buttons. The main configuration area is divided into sections:

- Associate:** "local register # 25" and "named csiActuator1" with the text "with this IoT attribute."
- Publish:** Includes checkboxes for "Using QOS", "Ack not required" (selected), "Ack required", and "Publish as" with options "Reported" (selected) and "Desired".
- Subscribe:** Includes a checked checkbox, "To topic index: 0", and the topic "v1/devices/me/rpc/request/+". Below this, it says "Apply this default value: 0.000000 after 0 minutes without any update from the cloud."

The URL displayed in the address bar when viewing your dashboard is accessible from your mobile device if the dashboard and corresponding device have been set to "make public". Use public viewing only for demo purposes. You will want to use more secure methods for production use.



If at any time you want to go back and make changes to widgets already on your dashboard, go into edit mode on your dashboard. Note that a pencil icon shows up on top of every widget. Click the pencil icon to make changes to the respective widget.



17.6 Diagnostics

To access diagnostics for your device, go to the Devices page and click on State (i.g. Active in this illustration).

Devices		Device Filter				
<input type="checkbox"/>	Created time ↓	Name	Device profile	Label	State	Customer
<input type="checkbox"/>	2024-03-12 09:20:43	VP6-1460	default	Test Device VP6-1460	Active	

You can view latest telemetry as illustrated previously. This will be any data published to the `v1/devices/me/telemetry` topic. You can also view the Audit logs.

VP6-1460

Device details

← butes Latest telemetry Alarms Events Relations **Audit logs** Version control >

🕒 last day 🔁 🔍

Timestamp ↓	User	Type	Status	Details
2024-03-12 11:08:25	jimhogenson@csimn.com	RPC call	Success	...
2024-03-12 11:08:24	jimhogenson@csimn.com	RPC call	Success	...
2024-03-12 11:08:23	jimhogenson@csimn.com	RPC call	Success	...

The Audit logs show all RPC calls that were involved in making any control widgets work. Click on the Details icon to see the entire exchange between Thingsboard and our MQTT device.

Audit log details

✕

Action data

```
{
  "entityId": "b5d8fd40-e07b-11ee-bc03-55147b896",
  "oneWay": true,
  "method": "set_csiActuator1",
  "params": "true"
}
```

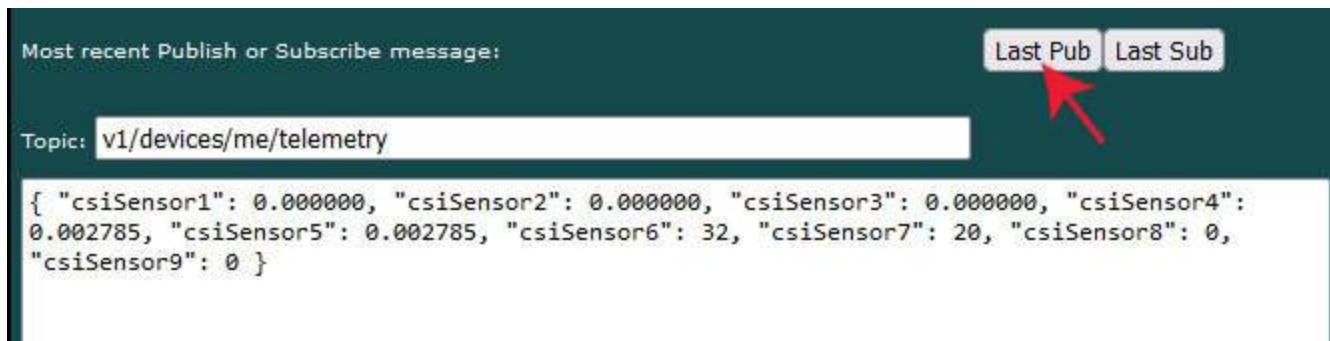
Close

The diagnostics available on the MQTT device side are the same as for any other MQTT service. Go to the Test page and click on Last Pub to see what was most recently sent by the device to the server (in this case Thingsboard).

Most recent Publish or Subscribe message: Last Pub Last Sub

Topic: `v1/devices/me/telemetry`

```
{ "csiSensor1": 0.000000, "csiSensor2": 0.000000, "csiSensor3": 0.000000, "csiSensor4": 0.002785, "csiSensor5": 0.002785, "csiSensor6": 32, "csiSensor7": 20, "csiSensor8": 0, "csiSensor9": 0 }
```

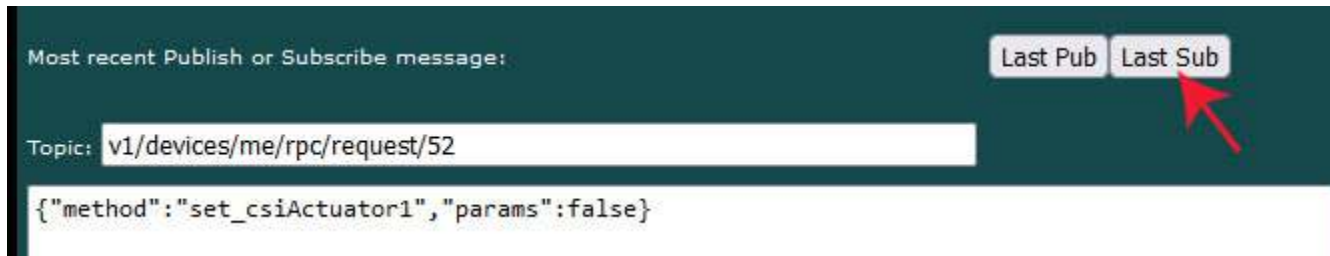


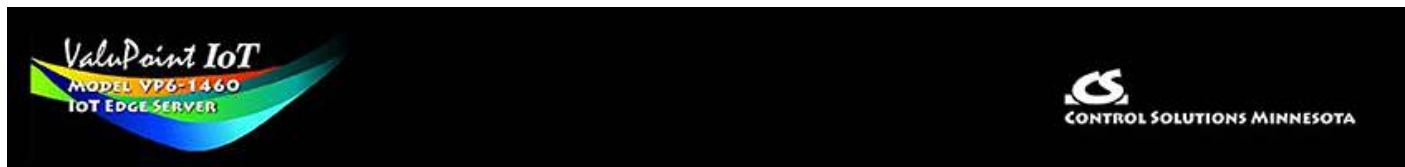
Click on Last Sub to see what most recently came from the server.

Most recent Publish or Subscribe message: Last Pub Last Sub

Topic: `v1/devices/me/rpc/request/52`

```
{"method": "set_csiActuator1", "params": false}
```





18. Programming with Script Basic

18.1 Creating a Program

You can use any plain text editor to create a program file on your PC, and then upload that file via the File Manager. Using a PC-based text editor is recommended especially for larger programs. The ValuPoint includes a very simple text editing page under the View/Edit tab, and this is suitable for creating small programs or making minor changes. An external plain text editor is recommended for larger programs.

To create a new program, enter a new file name ending in ".sb" for your program. The program should use only alphanumeric characters and be limited to 20 characters. As soon as you click the New button, the file will be created and automatically selected. If you are returning to edit a previously created program, select that file from the File list, and click Select.



There is a local, very simple text editor available via the web View/Edit page. To edit an existing file, start by clicking Get. After entering a new program, or editing an existing program, click Save.

It is recommended that you use an external text editor for large programs, and simply upload it on the Program File page.

The Language Help link provides a summary of Script Basic. For a complete reference,

use the external compiled help file available for Script Basic.



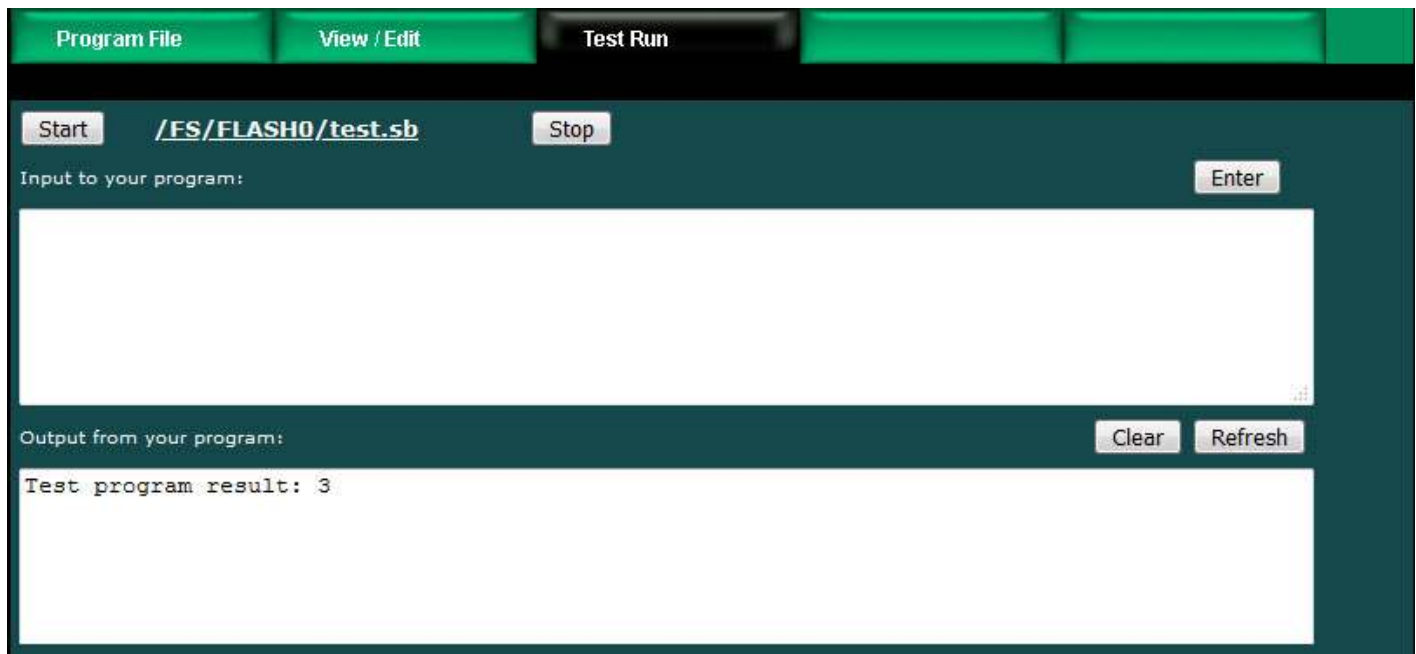
18.2 Testing the Program

The virtual terminal on the Test Run page can be used while testing programs. Any "print" statement will send its output to the Output window, and any "line input" statement will take input from the Input window. The print and line input statements will have no effect when running in the background (i.e. running as Auto run from startup).

The Virtual Terminal page does not auto refresh, therefore any output produced by a print statement will not appear until you click the Refresh button. Some initial output may appear immediately since the program began running faster than the initial page refresh that occurs after clicking Start, but you will need to click Refresh to see additional output.

WARNING: If you are testing a program, you should select No Auto (see below), then restart the ValuPoint. You will have two programs running at the same time if the auto run program is running and you are also running a program here. The results can be unpredictable if you are running two copies of the same or similar program at the same time.

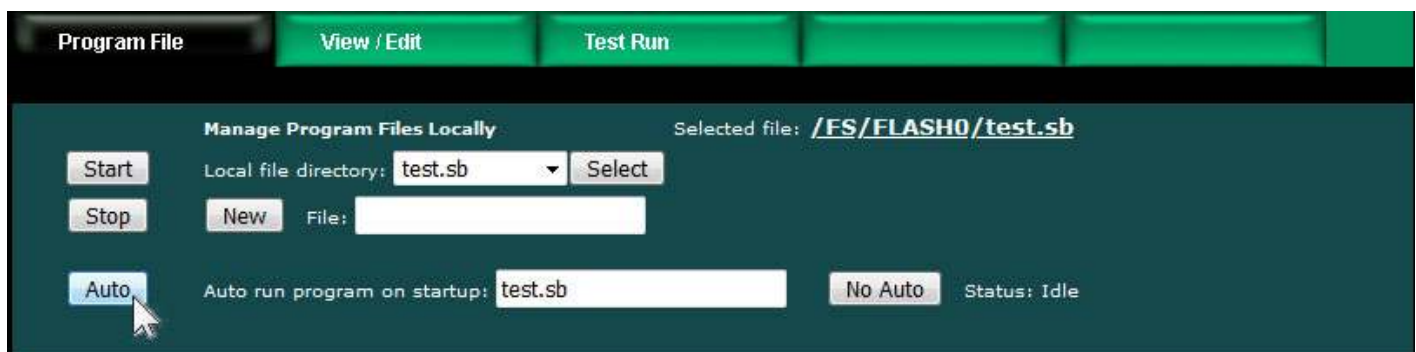
An abbreviated screen shot of the virtual terminal is illustrated below. In this example, output from the above simple test program is illustrated.



18.3 Setting the Program to Auto-Run on Startup

Your program will not be very useful if it does not start up when the ValuPoint boots up. You cause that to happen by selecting your file from the list, and then clicking the Auto button. Setting the auto run program will take several seconds as Flash memory is being reprogrammed at this point.

From this point on, your program named in the "Auto run program on startup" window will be automatically started upon bootup. Of course, if the program has errors, it might not keep on running. Be sure to test your program ahead of time, and also consider use of the "On Error" feature of Basic. What exactly you might do upon error is entirely up to you, but one potentially useful option is to set an error number of your own making in a specific data object that will be reported to the web portal using "Report on Delta".



To eliminate the auto run program, simply click the No Auto button. Clicking the No Auto button will only make certain no program is started upon bootup from this point forward. You will need to restart the ValuPoint to halt any program that was already running from the last bootup.

18.4 Special Functions

18.4.1 Serial Port Functions

Opening Comm Port:

To open the communication port as file #2 for example, you would use:

```
open "COM:9600" for comm as 2
```

The line end character is otherwise known as line feed or "\n". The carriage return will be ignored, unless it is specified to be the line end instead. To open the comm port using the carriage return as line end instead of the Linux line end, you would use:

```
open "COM:9600,CR" for comm as 2
```

Many devices return both carriage return and line feed at the ends of lines. The sole line end character recognized as the end of line for the comm port will end the line while the other will be discarded and not returned in the string that gets placed in a variable for Basic.

Any of the baud rates valid for typical serial port usage may be specified in place of the 9600 baud used in the above examples. The baud rate may be anything from 1200 to 115,200.

```
open "COM:9600,EVEN" for comm as 2  
open "COM:9600,ODD" for comm as 2  
open "COM:9600,2STOP" for comm as 2
```

Port settings will default to no parity and one stop bit (8 data bits). You can change parity by using the notations illustrated above. You may still add ",CR" to the string when parity is included.

Input from Comm Port:

The "line input" in Basic is used to receive entire lines from the comm port. For example,

```
line input #2, myLine
```

will accept a line up to the line end character into the variable myLine.

The "line input" in Basic expects to receive a full line terminated by a line end character. If you want to capture one character at a time and not be concerned with whole lines or line end characters, the following exmple illustrates capturing one character at a time and outputing one character at a time to the virtual terminal screen, until that one character is a carriage return - then the program closes the port and termiantes in this simple example.



Output to Comm Port:

Any variation on the file print referencing the comm port file number will send output to the comm port. For example,

```
print #2, "You typed ",myLine,"\n"
```

will echo the line received above right back to the comm port along with the comment 'You typed '.

Comm Port Timeout:

```
timeout (n)
```

Sets timeout in seconds that the "line input #n" request for input from the communication port will wait before returning an empty string if nothing was received on the communication port. Otherwise, the line input will wait indefinitely for a line that ends in a line end character.

18.4.2 Register Access

You may read any of the local Modbus registers using the getreg function, and write them using the setreg command. Note that getreg is a function while setreg is not; it is a command and therefore requires no parenthesis.

Usage is:

```
data = getreg (x)
setreg x, data
```

where 'x' is the Modbus register number, and 'data' is the value contained in that register. Consider the following example:

```
MyData = getreg (22)
MyData = MyData * 2.5
setreg 24, MyData
```

The above example will get the value of register 22 and place it in the variable MyData, then multiply it by 2.5, then place the resulting value in register 24. Variables may be used in place of the constants used for register numbers in the above example.

18.4.3 Register New-Data Status

```
n = getregstatus (x)
```

Check status for register number 'x', will return 1 if new data since last call, 0 if no new data, -1 if given register number is invalid. New data means the register's data value was changed by something other than this program, e.g., some external Modbus device wrote a new value to this register.

18.4.4 Register Access Status

You may optionally check to see if your most recent getreg or setreg call resulted in finding a valid register number. Simply use the getreg function with a register number of zero to check the previous getreg or setreg call.

```
n = getreg (0)
```

The value of 'n' will be 1 if the previous operation was successful, or 0 if it failed.

18.4.5 LED Control via Phantom Register Access

A set of "phantom" registers exists for the purpose of allowing your Script Basic program to turn on and off the Request and Reply LEDs at will. There are two sets of registers. One group will cause an automatically timed "flash" of the respective LED while the other group will provide static on/off control of the LEDs. Static on/off means once turned on, it will remain on until you explicitly turn it off again. Flash and static cannot both be used at the same time. The LEDs in the BB3-6101 are bi-color, meaning they can each be turned on to one of two colors, but of course not at the same time.

The phantom registers are only accessible from Script Basic, but use the same setreg command that is used to place values into the local Modbus registers (or more correctly, data objects accessible as Modbus registers). To "flash" an LED, write a non-zero value to the respective register. For static control, write a non-zero value to the static register to turn the LED on, and write zero to the same register to turn the LED off.

Register No.	LED function
10001	Request LED Flashes Green (N/A on MQ-61)
10002	Request LED Flashes Yellow (MQ-61 Yellow)
10003	Reply LED Flashes Green (MQ-61 Green)
10004	Reply LED Flashes Red (MQ-61 Red)
10005	Request LED Green Static On/Off (N/A on MQ-61)
10006	Request LED Yellow Static On/Off (MQ-61 Yellow)
10007	Reply LED Green Static On/Off (MQ-61 Green)
10008	Reply LED Red Static On/Off (MQ-61 Red)

For example, to flash the Reply LED Red, you would use:

```
setreg 10004,1
```

The LED registers can be written but not read. Using getreg on the LED registers will not return the LED state - your program needs to remember what it did.

18.4.6 Program Watchdog Timer

A watchdog timer is available to Script Basic via the following phantom register.

Register No.	Watchdog function
11000	Set/Reset Watchdog Timeout in Seconds (zero to disable)

The following example will set the watchdog timer to half a second.

```
setreg 11000,0.5
```

If your program does not call setreg to set/reset the watchdog timer again within this amount of time, the ValuPoint will be restarted (as if power cycled).

18.4.7 Error Information Retrieval

```
n = geterr (x,y)
```

Returns n=0 for no error, otherwise returns error code or status for item specified by 'x', item number 'y' (1..N). Error codes returned are those displayed on the respective web pages as error codes or device status.

Item 'x' may be:

4 = Modbus TCP device

5 = Modbus RTU device

10 = Modbus TCP read map (error * 100 + exception)

11 = Modbus TCP write map (error * 100 + exception)

12 = Modbus RTU read map (error * 100 + exception)

13 = Modbus RTU write map (error * 100 + exception)

Modbus TCP device code returned will be the connection status as listed in the Quick Help section of the Modbus TCP Devices page in the ValuPoint.

Modbus RTU device code will simply be an indication of whether any errors have been tabulated for that slave address, with the returned code indicating as follows:

1 = 'No response' errors have been tabulated

2 = CRC errors have been tabulated

3 = Exception errors have been tabulated for this RTU device.

Modbus 'error' for read/write maps is as follows, with exception being as defined by Modbus protocol when 'error' is Exception. Exception codes are as defined for Modbus protocol.

1 = TCP transaction ID mismatch

2 = Exception (see exception codes)

- 3 = Function code mismatch
- 4 = Insufficient data received
- 5 = No response (time-out)
- 6 = CRC error
- 9 = Host time-out

18.4.8 IP Address Retrieval

```
ip$ = getipaddr (x,y)
```

Returns IP address as a string for item type 'x', item number 'y' (1..N), formatted as character string.

Item 'x' may be:

- 0 = Our own IP address
- 4 = Modbus TCP device

If x=0 then y is ignored. If x=4, then y is device number on Devices page in Modbus TCP Setup.

18.4.9 Free Memory

You can find out within your program how many bytes of free memory remain. In the line below, 'n' will be the number of bytes. It will typically start out over 1,500,000. When it drops to near zero, the system will stop functioning because memory allocation from the heap has run up against stack space.

```
n = freemem()
```

This function should only be used for diagnostics while developing your program, and not be used in a production program that is expected to run indefinitely.

18.5 Example: Program Triggering of Publish to AWS

The ValuPoint provides support for character strings as a series of Modbus registers that are treated as a single piece of data just like 32-bit integers are a pair of registers treated as one. The most powerful aspect of this feature is the ease of creating and manipulating character strings in Script Basic. Consider the simple program illustrated here.

We start by creating a character string "register", which is really a series of 15 Modbus registers treated as one single data item.

We have configured an IoT Thing Point (Attribute) to publish to the custom topic that you would set up in the Simple Notification Service on the AWS side. By setting the rule to publish on "change by zero", we are telling the ValuPoint to publish any update of the local register.

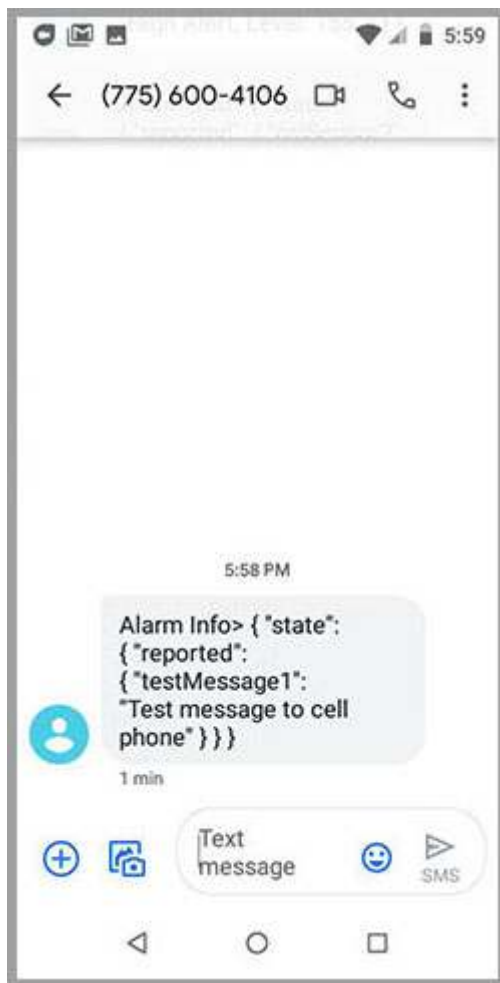


A useful program would be more complex than what is illustrated here. But this simple example does illustrate the minimum requirement for a Script Basic program causing a message to be sent to a mobile phone using AWS Simple Notification Service.

The program is illustrated on the View/Edit page. To run the program, you would go to the Test Run page and click Start.



Upon running the test program, the publish message can be viewed on the Thing Status :: Test page to verify that it was sent.



The combination of the Simple Notification Service and the simple test program illustrated here results in the text message being displayed on the mobile phone.

IMPORTANT: This demo was initially created prior to the cellular service industry implementing tighter rules intended to cut down on spam texting. While SNS does still work for sending text messages, you will need to register your 10 digit "from" number before being allowed to generate text messages from SNS.

18.6 Example: Writing to an Alphanumeric Display

As noted in the previous example, the ValuPoint provides support for character strings as a series of Modbus registers that are treated as a single piece of data just like 32-bit integers are a pair of registers treated as one. The most powerful aspect of this feature is the ease of creating and manipulating character strings in Script Basic. Here is another example.

The screenshot shows the ValuPoint IoT web interface. At the top left is the logo for ValuPoint IoT, MODEL VP6-1460, IOT EDGE SERVER. At the top right is the logo for CONTROL SOLUTIONS MINNESOTA. Below the logos is a navigation menu with buttons for Local Data, Modbus, IoT Cloud, System, System Setup, Programming, and Scheduler. The Programming button is highlighted. Below the navigation menu are buttons for Program File, View / Edit, and Test Run. The View / Edit button is highlighted. Below the buttons is a text area containing the following script:

```
setreg 21, "Hello from Basic"  
setreg 41, "This is a demo"  
setreg 61, "of writing display"  
setreg 81, "from program."
```

At the bottom of the text area are buttons for Page Down, Page Up, Language Help, Get, and Save. To the right of the buttons is a file path: File: /FS/FLASH0/display1.sb.

We are going to send this 4-line message to an alphanumeric display that has a Modbus RTU interface. To start out, we need to configure the Modbus RTU port for the baud rate that matches the display. The ValuPoint will be the master since the display is an RTU slave.

The screenshot shows the ValuPoint IoT web interface with the Modbus configuration page. The navigation menu is the same as in the previous screenshot. The Modbus button is highlighted. Below the navigation menu are buttons for RTU Setup, RTU Data, TCP Setup, and TCP Data. The RTU Setup button is highlighted. Below the buttons are buttons for Local Device, RTU Read Map, and RTU Write Map. The RTU Write Map button is highlighted. Below the buttons is an Update button. The main content area contains the following configuration options:

- Baud Rate: 9600
- Parity: None, 1 Stop Bit
- I am the Master:
- I am a Slave:
- Parameters for RTU Master:
 - Default Poll Rate: 5.000 Seconds
 - Timeout: 1.000 Seconds
- Parameters for RTU Slave:
 - My Address or Unit #: 0
- Use FC 5/6 instead of 15/16 for unit numbers (slave addresses) starting at 0

Next, we create a set of 4 Modbus RTU Write Maps, one per line since the 4 lines are treated as 4 Modbus registers here.

Local Data		Modbus		IoT Cloud		System	
RTU Setup		RTU Data		TCP Setup		TCP Data	
Local Device		RTU Read Map		RTU Write Map			
Showing 1 to 5 of 5							
<input type="button" value="Update"/> <input type="button" value=" < Prev"/> <input type="button" value=" Next >"/>							
Map #	Local Register #	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Name	
<u>1</u>	21	Holding Register	CHAR	1	1	Char String 1	
<u>2</u>	41	Holding Register	CHAR	11	1	Char String 2	
<u>3</u>	61	Holding Register	CHAR	21	1	Char String 3	
<u>4</u>	81	Holding Register	CHAR	31	1	Char String 4	
<u>5</u>	0	None	None	0	0		

When creating character string maps, it is necessary to use the expanded form to enter the map. Click on the Map number in the first column to access the expanded form for each register. Once you have the expanded form showing for the first register, you can simply click Next to move to the next one.

The one bit of configuration important to character strings that must be entered here is the number of characters. Note that this is number of characters, not number of Modbus registers. The ValuPoint automatically stores two ASCII characters per 16-bit holding register. Therefore, our 20-character strings will occupy 10 registers in Modbus address space.

Note that "when local register changes by" is functional for character registers. The "greater than" value should be left set at zero so that any change results in updating the display. Any attempt at a numeric comparison with a string for purposes of update will not succeed.

Local Device	RTU Read Map	RTU Write Map		
Map # <input type="text" value="1"/>				
<input type="button" value="Update"/> <input type="button" value=" < Prev"/> <input type="button" value=" Next >"/>				
Read local register # <input type="text" value="21"/> named <input type="text" value="Char String 1"/>				
Write remote register <input checked="" type="checkbox"/> when local register changes by > <input type="text" value="0.000000"/> or <input checked="" type="checkbox"/> when <input type="text" value="10.0"/> seconds have elapsed with no change.				
Otherwise write remote register unconditionally, applying local register data as follows:				
Apply scale: <input type="text" value="0.000000"/> and offset: <input type="text" value="0.000000"/> Then if applicable, apply bit mask: <input type="text" value="0000"/> and bit fill: <input type="text" value="0000"/>				
Write <input type="text" value="Holding Register"/> as <input type="text" value="Character String"/> Size: <input type="text" value="20"/> with blank padding if checked <input checked="" type="checkbox"/>				
To register # <input type="text" value="1"/> at Unit # <input type="text" value="1"/> With low register first if checked: <input type="checkbox"/>				
Repeat this process <input type="radio"/> at least <input type="radio"/> no more than every <input type="text" value="0.0"/> seconds.				
<input type="checkbox"/> Enable this map only when index register <input type="text" value="0"/> is set to a value of <input type="text" value="0"/>				
# Client Write Maps Enabled: <input type="text" value="5"/>				
<input type="button" value="Insert"/> <input type="button" value="Delete"/>				

If we click Start on the Programming Build/Run page, the character registers will be set as illustrated below.

Local Registers		Calculate	Copy	Report	
Showing registers from				1	Update < Prev Next >
Local Register #	Register Name	Set	Register Data	Register Format	
00001	Data Value 1	<input type="checkbox"/>	0.000000	Single Float	
00003	Data Value 2	<input type="checkbox"/>	0.000000	Single Float	
00005	Data Value 3	<input type="checkbox"/>	0.000000	Single Float	
00007	Data Value 4	<input type="checkbox"/>	0.000000	Single Float	
00009	Data Value 5	<input type="checkbox"/>	0.000000	Single Float	
00011	Data Value 6	<input type="checkbox"/>	0.000000	Single Float	
00013	Data Value 7	<input type="checkbox"/>	0.000000	Single Float	
00015	Data Value 8	<input type="checkbox"/>	0.000000	Single Float	
00017	Data Value 9	<input type="checkbox"/>	0.000000	Single Float	
00019	Data Value 10	<input type="checkbox"/>	0.000000	Single Float	
00021	Char String 1	<input type="checkbox"/>	Hello from Basic	Char String[40]	
00041	Char String 2	<input type="checkbox"/>	This is a demo	Char String[40]	
00061	Char String 3	<input type="checkbox"/>	of writing display	Char String[40]	
00081	Char String 4	<input type="checkbox"/>	from program.	Char String[40]	

Assuming the Modbus mapping has been set up as illustrated above, the alphanumeric display will now display the message generated by Script Basic. This particular inexpensive display is the model SC2004MBS-B with RS485 interface available at siliconcraft.net. If you set the Modbus RTU port parameters to 9600 baud, and set write maps to send data to slave address (unit) 1, no configuration of the display is needed out of the box.



18.7 Example: Device and Rule Diagnostics

Here is a short example illustrating the special diagnostic functions available in Script Basic. In this example, we are checking on our ValuPoint's attempts to query a Modbus TCP device. The device is #1 in the Modbus TCP device list, and the rule we are checking is Modbus TCP Read Map #1.

The screenshot shows a programming IDE with a menu bar (Program File, View / Edit, Test Run) and a toolbar (Page Down, Page Up, Language Help, Get, Save). The file path is /FS/FLASH0/testError.sb. The script content is as follows:

```
ip$ = getipaddr ( 4, 1 )
print ip$, "\r\n"
code = geterr ( 4, 1 )
print code, "\r\n"
code = geterr ( 10, 1 )
print code, "\r\n"
```

Normal results will be to display the IP address and two error codes that will be zero if there are no problems.

The screenshot shows the output window with the text "Output from your program:" and buttons for "Clear" and "Refresh". The output is:

```
192.168.1.109
0
0
```

Error 202 from the read map says we are receiving an exception code from the Modbus TCP slave. In this case, we deliberately tried to read a holding register when the slave only had input registers available, for illustration purposes.

The screenshot shows the output window with the text "Output from your program:" and buttons for "Clear" and "Refresh". The output is:

```
192.168.1.109
0
202
```

This example was created using ModSim as the Modbus TCP slave. By terminating the ModSim program, we no longer have any Modbus TCP device accessible. This results in a timeout indicated by the read map (500) . The device status is indicating 111, which means the IP address is reachable, but Modbus TCP cannot be found there - and this is the expected error in this deliberately flawed example.

The screenshot shows the output window with the text "Output from your program:" and buttons for "Clear" and "Refresh". The output is:

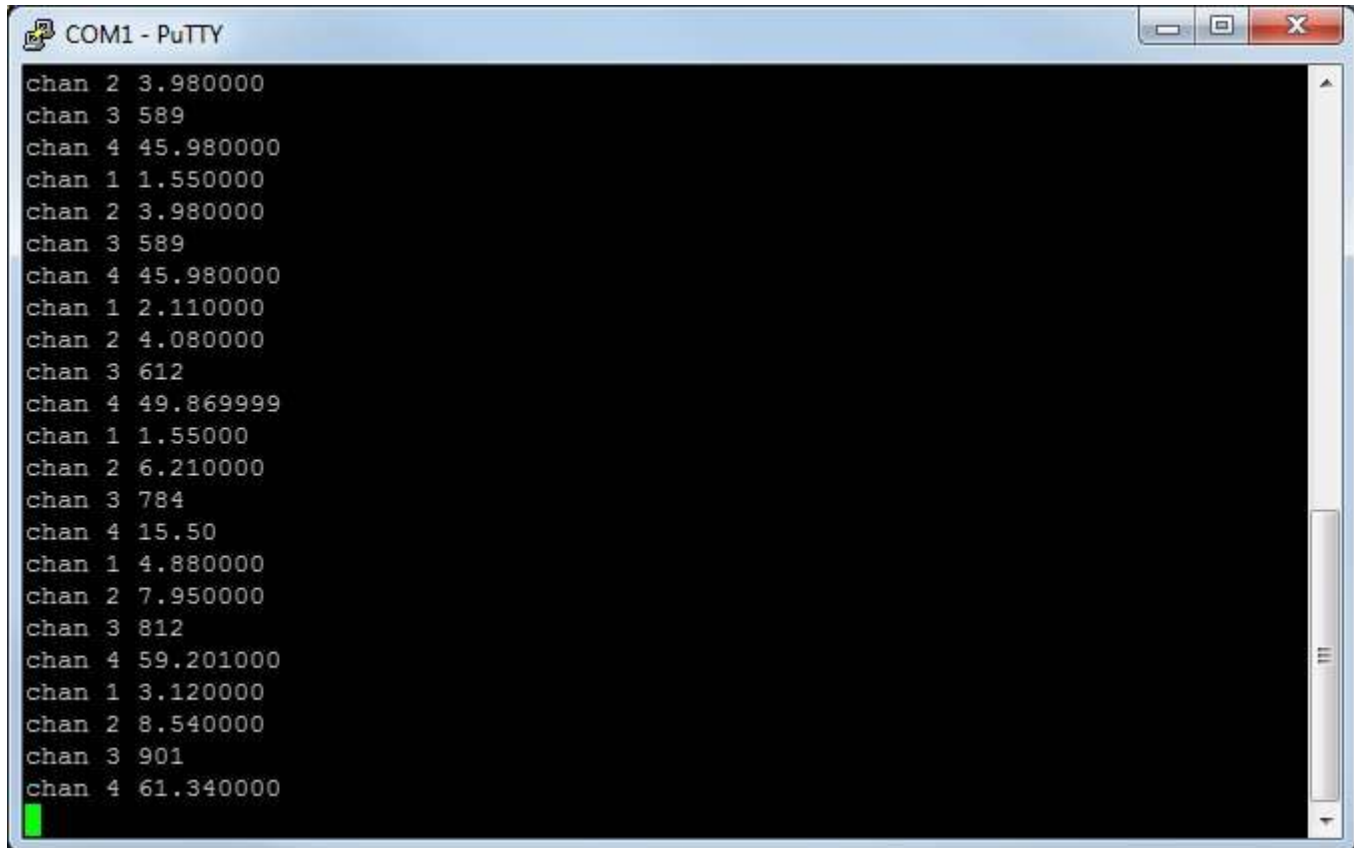
```
192.168.1.109
111
500
```

18.8 Example: Connecting Custom Serial Device to AWS

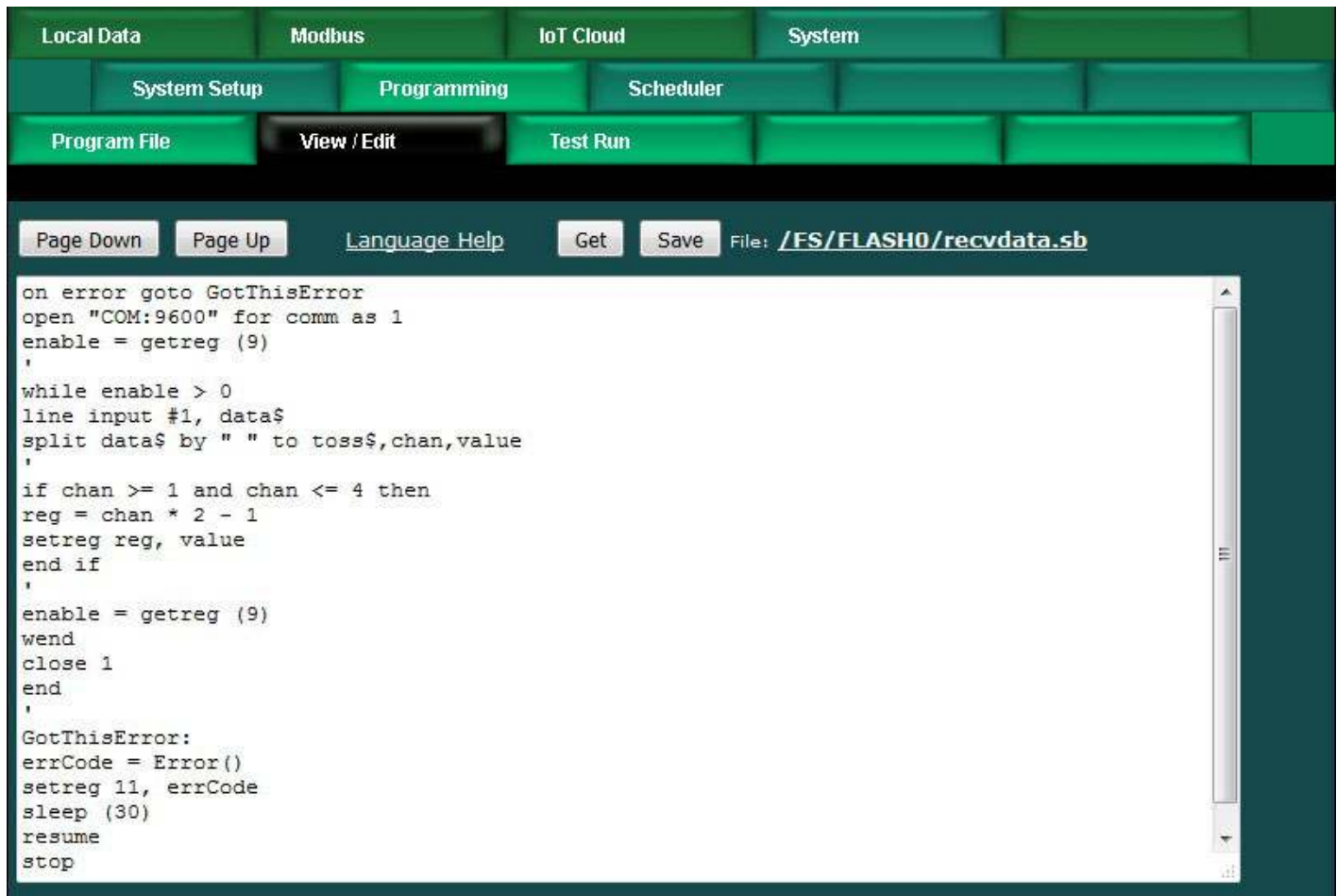
A very useful feature available with the ValuPoint is the ability to disable Modbus RTU and use the serial port for a proprietary serial protocol instead. When Modbus RTU is disabled, the baud rate of the port is set by Basic. The proprietary protocol may be one where you need to send commands to the device in order to receive responses, or it may be a simple serial data logger that periodically sends character strings that you wish to capture.



Our first example will capture data from a data logger type device that periodically sends strings of data. In this case, it is simply a channel number and that channel's data value. This device is a 4-channel device, and its output is illustrated here by simply connecting it to a PC (via RS485 to RS232 adapter) and running PuTTY to see its output.



The program to capture data from this device and store the results in local registers is illustrated below. The key line to notice here is the "split" where the data line is effectively parsed. The "toss\$" is going to end up holding the string "chan" which we don't care about. The numbers representing channel number and that channel's data value will end up in the variables "chan" and "data". Then, based on channel number, we calculate a register number. This is needed because we are going to store the data in floating point registers which occupy 2 Modbus registers worth of address space each.



The local register values are going to continue changing as new data is received. The screen shot below illustrates the most recently received data in this example.

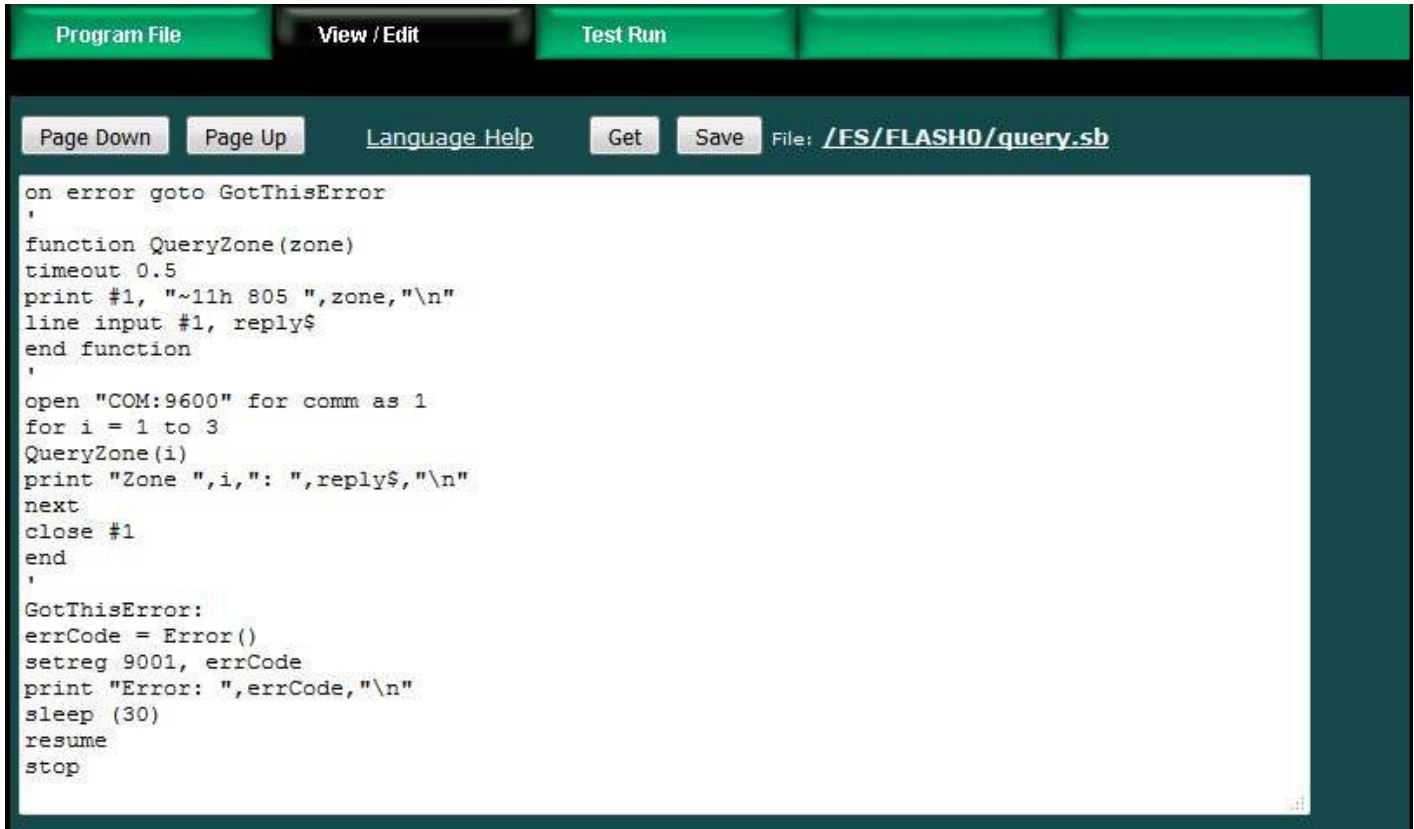
The screenshot shows the 'Local Registers' window with buttons for Calculate, Copy, and Report. It indicates 'Showing registers from 1' and has Update, < Prev, and Next > buttons. The table below displays the current register data:

Local Register #	Register Name	Set	Register Data	Register Format
00001	Data Value 1	<input type="checkbox"/>	3.120000	Single Float
00003	Data Value 2	<input type="checkbox"/>	8.540000	Single Float
00005	Data Value 3	<input type="checkbox"/>	901.000000	Single Float
00007	Data Value 4	<input type="checkbox"/>	61.340000	Single Float
00009	Enable Logger	<input type="checkbox"/>	1.000000	Single Float
00011	Data Value 6	<input type="checkbox"/>	0.000000	Single Float

Once data becomes available in the local registers, you can set up IoT Publish rules the same as you would for publishing Modbus data.

The above example is always just receiving data that is sent automatically. If you needed to query a device in order to receive data, you can do that from Script Basic. The following example shows querying one particular type of device. The string printed in the QueryZone function is unique to this device - you need to format whatever is unique to your device.

Note the "timeout" function. This causes the serial port driver to return an empty string if nothing is received after waiting half a second in this example. This allows your Basic program to continue on with other things if this particular device did not respond (e.g. got disconnected).



The screenshot shows a Basic programming IDE with a menu bar (Program File, View / Edit, Test Run) and a toolbar (Page Down, Page Up, Language Help, Get, Save). The file name is /FS/FLASH0/query.sb. The code in the editor is as follows:

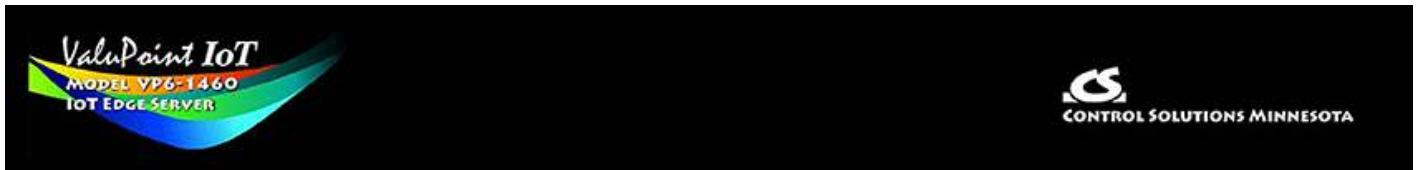
```
on error goto GotThisError
'
function QueryZone(zone)
timeout 0.5
print #1, "~11h 805 ",zone,"\n"
line input #1, reply$
end function
'
open "COM:9600" for comm as 1
for i = 1 to 3
QueryZone(i)
print "Zone ",i," ": ",reply$,"\n"
next
close #1
end
'
GotThisError:
errCode = Error()
setreg 9001, errCode
print "Error: ",errCode,"\n"
sleep (30)
resume
stop
```

The "line input" in Basic expects to receive a full line terminated by a line end character. If you want to capture one character at a time and not be concerned with whole lines or line end characters, the following example illustrates capturing one character at a time and outputting one character at a time to the virtual terminal screen, until that one character is a carriage return - then the program closes the port and terminates in this simple example.



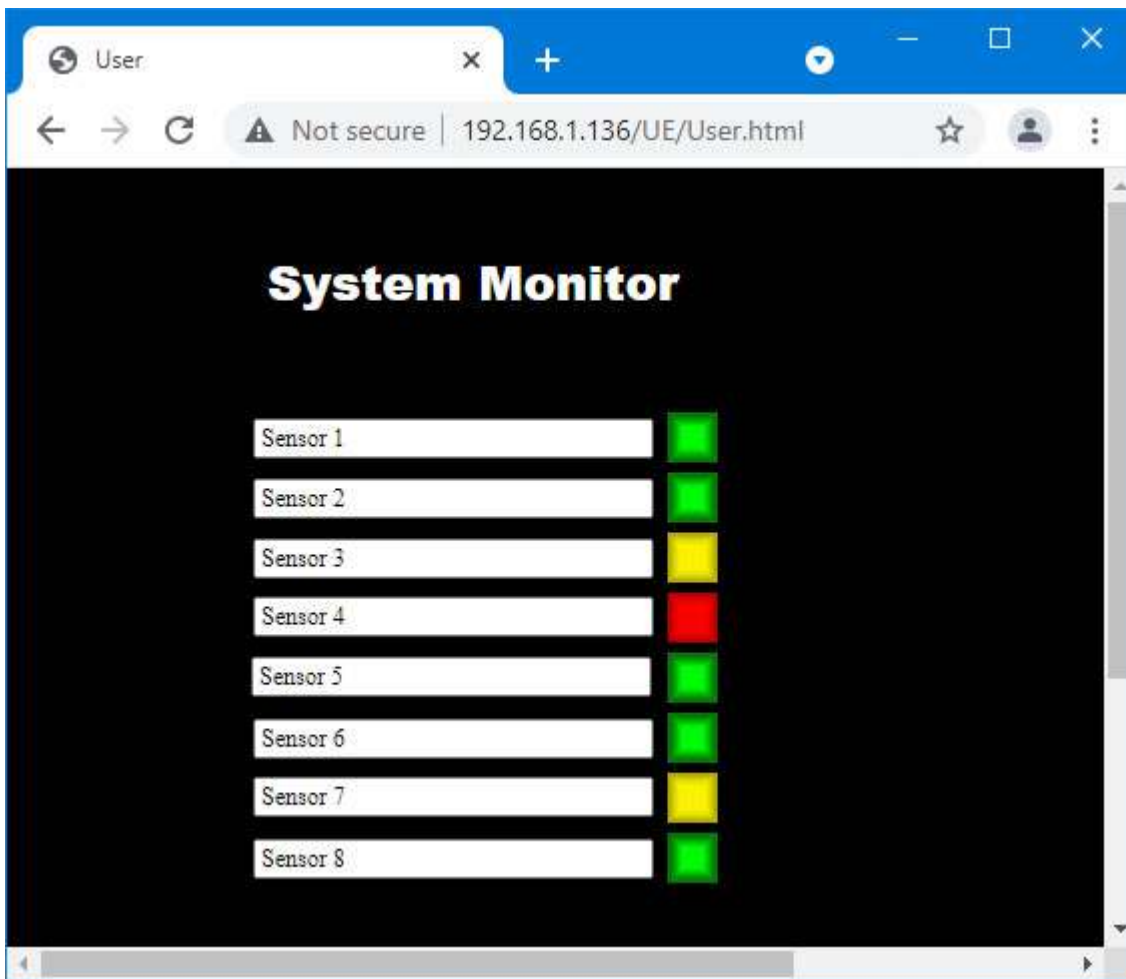
The screenshot shows a Basic programming IDE with a menu bar (Program File, View / Edit, Test Run) and a toolbar (Page Down, Page Up, Language Help, Get, Save). The file name is /FS/FLASH0/echo.sb. The code in the editor is as follows:

```
open "COM:9600" for comm as 1
repeat
a = input (1, 1)
print #1, a
until a = 13
close 1
```



19. User HTML

This section is an updated version of the user_html_cgi.pdf document found on the introduction page under User HTML Tutorials at <https://info.csimn.com>. This information has been updated per implementation in ValuPoint. Except for changing the link “../index.html” to “/html/index.html” where applicable, the various demos found in the knowledgebase should work in a ValuPoint.



The “naked pages” referenced in the 2008 version of the CGI overview are not implemented in ValuPoint. They were an attempt at allowing some level of “private labeling” the ValuPoint. If private labeling is desired, Control Solutions does offer that option which will result in fully rebranding the entire preprogrammed web UI.

19.1 Static HTML

User HTML may be installed as a “wrapper” around the default web pages. To install user HTML, use the File Manager (filter set to *.*) to upload any combination of .html,

.txt, .gif, and .jpg files. You can also include .pdf, .xml, .css and JavaScript .js files. Note that PHP, ASP, etc., are not supported.

You can also use FTP to upload files to the /FLASH0 (that's flash zero) directory and this will be the default directory upon opening an FTP session. An FTP session in Linux, Windows command prompt, or Windows PowerShell work well. Some smart FTP clients work, but some try to be too helpful and simply screw things up. We tested FileZilla at one point and it seemed to work ok, but we do not keep testing every revision of FileZilla.

The top level user file must be named User.html (case sensitive). If this file is present in /FLASH0, it will be served instead of the default index.html page pre-programmed in the device any time you simply browse to the device's IP address. Once this page is open, it may link to any other html files in the /FLASH0 directory. All user HTML is filtered as it is served to provide dynamic content.

There are a handful of tricks that must be observed to make user html work. All references to other user pages and to user image files must have the file names preceded with /UE/ as in /UE/User2.html or /UE/mypicture.gif. The UE stands for "user escape" and is treated as a virtual directory that actually points to /FLASH0. All preprogrammed pages are found in /html/ and preprogrammed images are found in /img/.

19.2 Dynamic Data Tags in User HTML

Dynamic Data – Creating a Form

Dynamic access to register data is provided. Dynamic updates of register contents is also supported via the form post method. The form must be defined using the following tags:

```
<form id="UserForm" action="/UE/icanForm" method="post"
name="UserForm">

</form>
```

The submit button causing the post must be defined as:

```
<input type="submit" name="submitChange" value="Change">
```

Any submit buttons other than those recognized as noted here will simply result in a page refresh. Only the submit button named "submitChange" with a value of "Change" will result in parsing of the form data. Only a form with action as named above will be parsed.

If you want to redefine the appearance of the button, you can implement a graphic button by including an image as follows, and then including the JavaScript function as shown:

```

```



```
<script type="text/javascript"><!--
function sendMeAway() {
    document.UserForm.submitChange.value="Change";
    document.UserForm.submit();
}
//--></script>
```

Input Types: <input type="text">

Two types of data input are recognized by CGI processing of the user post: Text input and option select. The search string keyed upon for text is **<input type="text">** and the search string keyed upon for the select option is **<select name=**

A text input should be constructed as follows:

```
<input type="text" name="reg22" value="%d" readonly size="8">
```

The contents of the register number included in the name ("regX") will be displayed when the page is served, and the data will be taken dynamically from the register at that point in time, and again each time the page is refreshed. The data will be formatted using the C format string found in the value tag. Integer formats (%d, %04d, %x, etc) should be used for integer registers, and floating point (%f, %.2f, etc) should be used for floating point registers. If "readonly" is specified, data will only be displayed in this window. Otherwise the data returned by the post will be parsed, and the result placed back into the register.

The following keywords are recognized as text input "names":

- regX – references the value in local register number X
- namX – references the name of register number X
- site – references the Thing Name (Thing ID page)

All of these data elements may be read, and will be written unless you specify "readonly". The definition of read means take data from the local register when serving the page, and write means write data to the local register if the form was submitted by the appropriately named submit button (see Form above).

Input Types: <select>

An option select should be constructed as follows:

```
<select name="reg25" size="1">
    <option selected value="0">OFF</option>
    <option value="1">ON</option>
</select>
```

The strings corresponding to the values given will be displayed when the register named matches that value, otherwise "---" will be displayed. When an option is selected and the form posted, the value corresponding to the new selection will be written back into the register. The "selected" tag shown above is not required since it is automatically inserted in the appropriate place (moved around) when the page is served.

Input Types: <input type="hidden">

An additional form of input has been added to filtered HTML. Hidden variables may be defined using the following syntax:

```
<input type="hidden" name="reg22" value="%d" readonly>
```

This will be processed the same as "text" input except the value is not displayed. This is useful as a means of providing non-displayed data to a JavaScript function. Hidden data upon return will be parsed and put back into registers unless `readonly` is specified. Omit `readonly` if hidden data should be parsed. This provides a means for JavaScript to get data back into registers.

Page Links

To create a link on the user page to get into the default preprogrammed pages, define a link to `"/html/index.html"`, for example:

```
<a href="/html/index.html">Log In</a>
```

To link to another user page in the FLASH0 directory, use a link such as:

```
<a href="/UE/pwUserP3.html">Room #1</a>
```

Links to graphic images you want shown on the page are created in similar fashion:

```

```

Note that you preface the page name with `"/UE/"` when the file is located in the FLASH0 directory, but preface the name with `"/html/"` when accessing a preprogrammed page.

Password Protection

There are 3 levels of password protection: Restricted, Maintenance, Administrator (root is a special form of administrator). User pages may be password protected at the "Restricted" level. To password protect a user page, simply insert the letters "pw" in front of the name. Therefore, if *User2.html* is a page you wish to protect, rename it *pwUser2.html*.

The top level page for User HTML must still be named *User.html* (and not *user.html* or *USER.html*). If you don't want anything useful to be completely unrestricted, simply put a plain dumb page in *User.html* with a link that says "log in" and link it to *pwUser.html*.

Note that "restricted" level of password protection means the user can access any "pw" user pages, but cannot access any pages beyond index in the pre-programmed page set.

Other Input Types

Radio buttons and other forms of input are not supported at this time. The HTML will be passed through, but not filtered and associated with register data. Therefore, you can use radio buttons, etc., with JavaScript, but you must explicitly associate the resulting data with hidden input variables in order to return the data to registers.

Additional Special Submit Buttons

The submit button causing the post for changing data values must be defined as:

```
<input type="submit" name="submitChange" value="Change">
```

Two additional button actions are available to save the configuration file or restart the device:

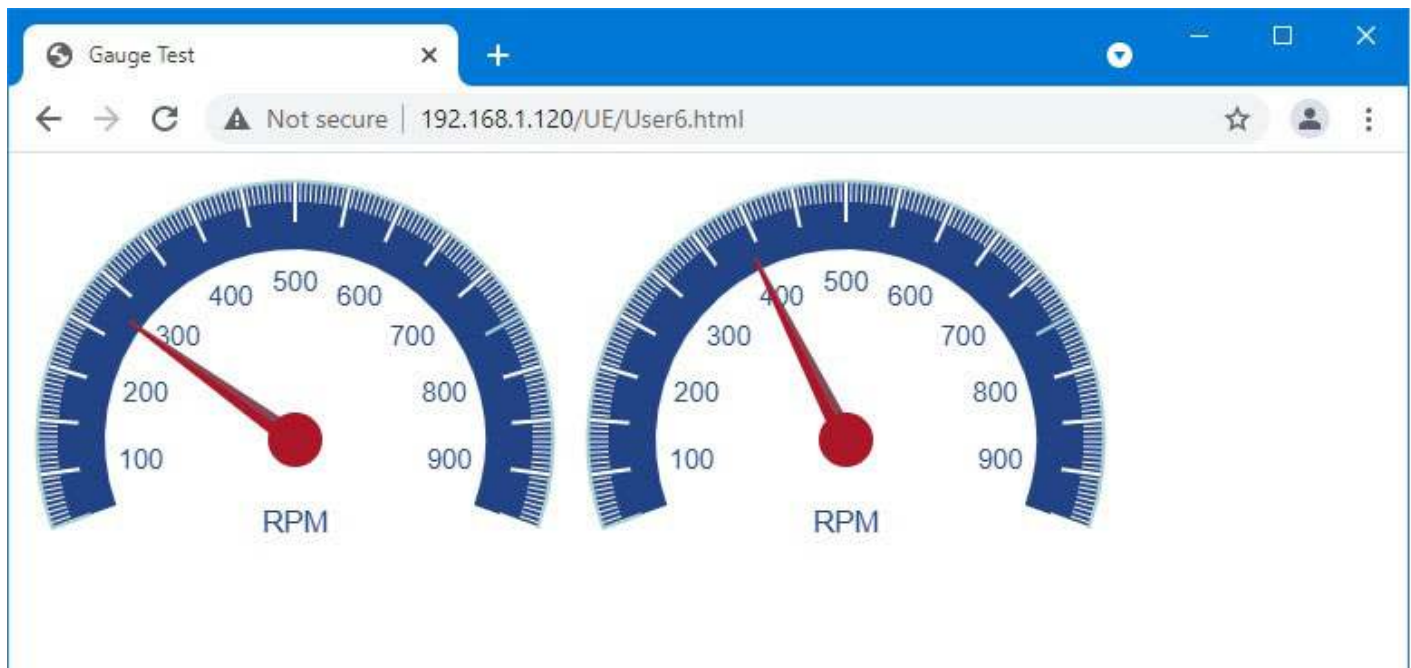
```
<input type="submit" name="submitSave" value="Save">  
<input type="submit" name="submitRestart" value="Restart">
```

19.3 Live JavaScript Gauges

An external JavaScript library can be specified. The JS file should be loaded into the /FLASH0 directory along with User.html, etc. In the HTML file, the script file is referenced as illustrated here by the first few lines of User.html that generated the gauges pictured in the screen shot.

```
<!doctype html>  
<html>  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
<title>Gauge Test</title>  
<link rel="stylesheet" href="/FS/FLASH0/fonts.css">  
<script src="/FS/FLASH0/gauge.min.js"></script>  
</head>  
<body style="background: #fff" onload="animateGauges()">  
  
<canvas id="canvasPressure"></canvas>  
<canvas id="canvasPressure2"></canvas>  
  
<script>  
var gaugePressure = new RadialGauge({
```

(entire project is available on csimn.com web site)



Many different JavaScript gauges are available at <https://canvas-gauges.com/>.

To cause the gauges to automatically update in real time, you need two things: The `animateGauges()` function in the JavaScript, and something that is going to retrieve real time data into the HTML document. The real time data retrieval is done by a hidden iframe, included in the User.html like this:

```
<iframe name="phantom" src="UserGetData.html" frameborder="0"
height="50" width="50"></iframe>
```

The UserGetData.html is constructed like this:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

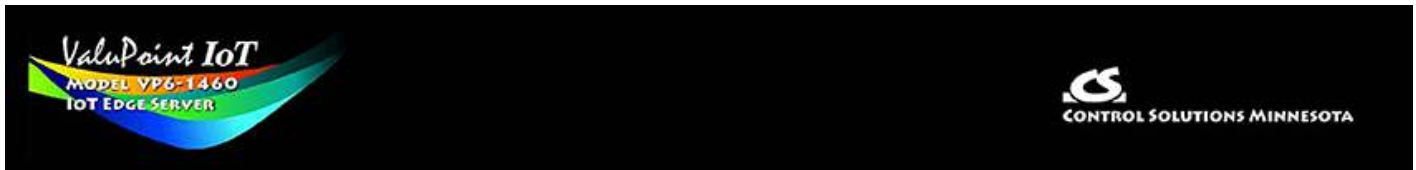
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
<title>Untitled Page</title>
<meta http-equiv="refresh" content="1">
</head>

<body bgcolor="#ffffff">
<form id="UserForm" action="/UE/icanform" method="post" name="UserForm">
<div align="left">
<input type="hidden" name="reg1" value="%d"><input type="hidden"
name="reg2" value="%d"></div>
</form>
<p></p>
</body>

</html>
```

NOTE: The above example using an iframe to retrieve real time data was created prior to adding the REST API capability to ValuPoint. The gauge animation could be

restructured to take advantage of the REST API if you know your way around JavaScript.



20. REST API

You can use the ValuPoint as a means of communicating with your Modbus devices using a REST API. The API must be enabled at the bottom of the Network setup page before the API will respond.

20.1 GET Data from Device

You can use an HTTP GET request to query the ValuPoint for its data values. The URL form of an acceptable query would be:

```
http://10.0.0.101/UE/query/csiSensor1
```

and this query will return a JSON format reply something like this:

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  "state": {
    "reported": {
      "csiSensor1": 47,
      "LocalTime": "2021-09-10T09:30:00-06:00"
    }
  }
}
```

The names given as register names on the Local Registers page will be used as data names here. The AWS IoT platform requires that names contain no embedded spaces, and avoiding spaces in names will also be necessary in most instances here. The data value will be either numeric as illustrated, or a character string enclosed in quotes. Timestamps conform to ISO 8601.

The same user name and password credentials otherwise required for web UI login will be required here. If the above query is done via a browser, the browser will ask for the username and password. When doing the query programmatically, the client must provide the credentials as applicable in the programming environment being used.

Example error response:

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  "error": "no such object"
}
```

The HTTP 200 OK refers to the validity of the HTTP protocol. It does not mean that your properly formatted query provided good data. You need to look at the JSON data to see if it contains "error". If you got HTTP 200 OK, the only two possible top level names are "error" or "state".

20.2 POST Data to Device

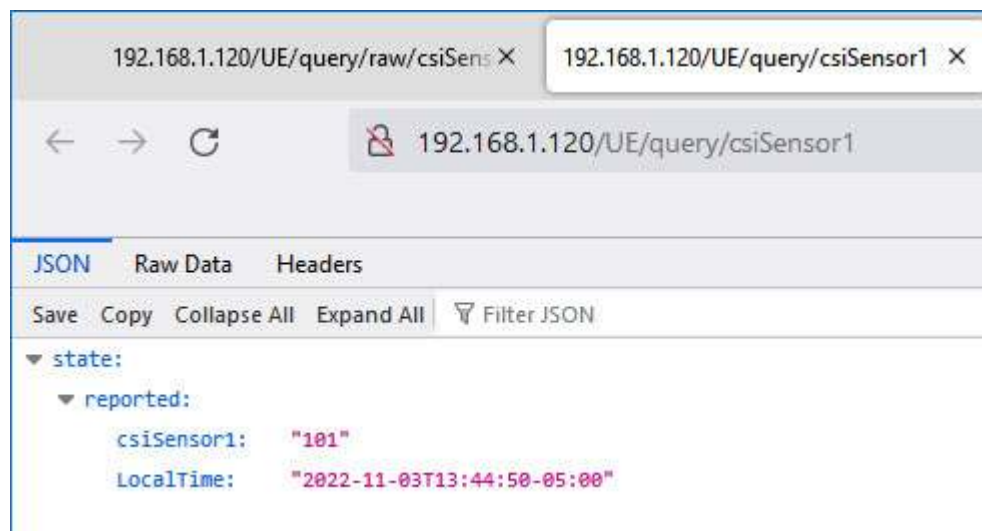
To write data to the ValuPoint (which will result in writing data to a Modbus register in some other device if the client is set up to do so), you can do an HTTP POST to `http://10.0.0.101/UE/query` with content such as:

```
{
  "state": {
    "desired": {
      "csiActuator1": 47
    }
  }
}
```

This example will set the register named `csiActuator1` to value 47 and return result similar to above GET reply with respective object name.

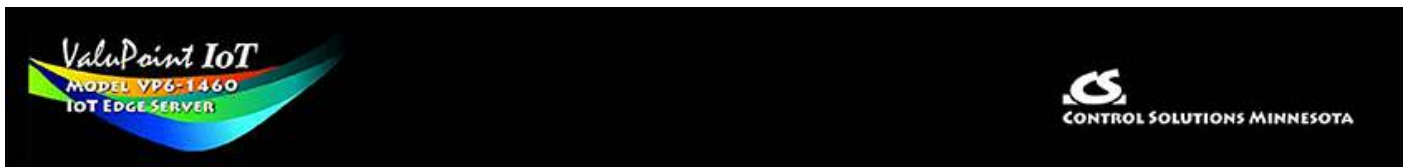
20.3 Raw Data Query

The normal REST API will return data as JSON formatted data as illustrated in the web browser screen shot below.



Some clients, however, are not able to parse JSON data and look for just raw data. The query now allows the "raw/" path to be inserted to obtain just raw data as illustrated below.





Appendix A Hardware Details

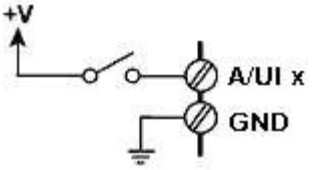
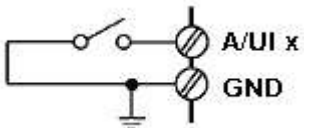
A.1 Wiring, Physical I/O Connections

Connection of Inputs

The VP6-1460 contains no configuration jumpers for configuring I/O points. There is no need to open the enclosure for configuration of I/O. Input types are switched under software control.

Input points should be connected as indicated in the various diagrams below. In addition to selecting a wiring diagram, the corresponding selections should be made on the I/O Config page as discussed in section 4.2 of this user guide.

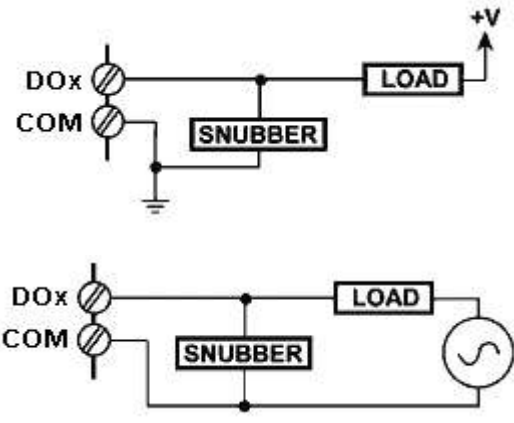
I/O Point Type	Wiring Guide	Additional Information
A/UI Analog Input: 0-10VDC Voltage Input or Pulse Count		<p>A/UI inputs 1-12 will accept voltage inputs of up to 10VDC. Voltages to 11VDC will be measured. Voltages to 24VDC will be tolerated, but measurement is internally limited to a reading of 11VDC.</p> <p>This wiring diagram is also applicable to pulse counter input for counting pulses from an active pulse generator.</p>
A/ UI Analog Input: 0-20mA Current Input		<p>A/UI inputs 1-12 will accept current inputs of 0-20mA with the addition of a 500 ohm 1/2 watt external resistor. A 500 ohm resistor will produce a 0-10VDC signal. A 250 ohm resistor may also be used to produce a 0-5VDC signal.</p>
A/UI Analog Input: Thermistor Input		<p>A/UI inputs 1-12 will accept thermistors of 3k, 5k, 10k, or 20k ohms. Linearization via interpolation of a 56-point table is performed internally.</p>

<p>A/UI or DI Discrete Input: Discrete Voltage</p>		<p>A/UI inputs 1-12 may be connected as discrete voltage sensing inputs. Inputs up to 24VDC are tolerated, but threshold sensing only functions over the 0-10VDC range.</p>
<p>A/UI or DI Discrete Input: Dry Contact Closure to Ground or Pulse Count</p>		<p>A/UI inputs 1-12 may be connected as discrete inputs sensing dry contact closure to ground. Internal excitation of 0.3mA is provided. The excitation current may be increased by the addition of an external resistor (pullup to DC power).</p> <p>This wiring diagram is also applicable to pulse counter input for counting pulses from a switch closure.</p>

Connection of Outputs

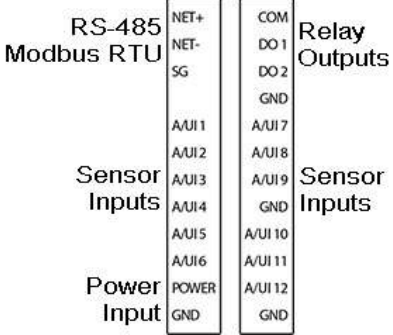
The VP6-1460 contains no configuration jumpers for configuring outputs. There is no need to open the enclosure.

Output points should be connected as indicated in the diagrams below. The discrete (relay) outputs are SPST N.O. with the common side connected together to the COM terminal. The COM terminal for relays is NOT electrically common to any of the GND terminals.

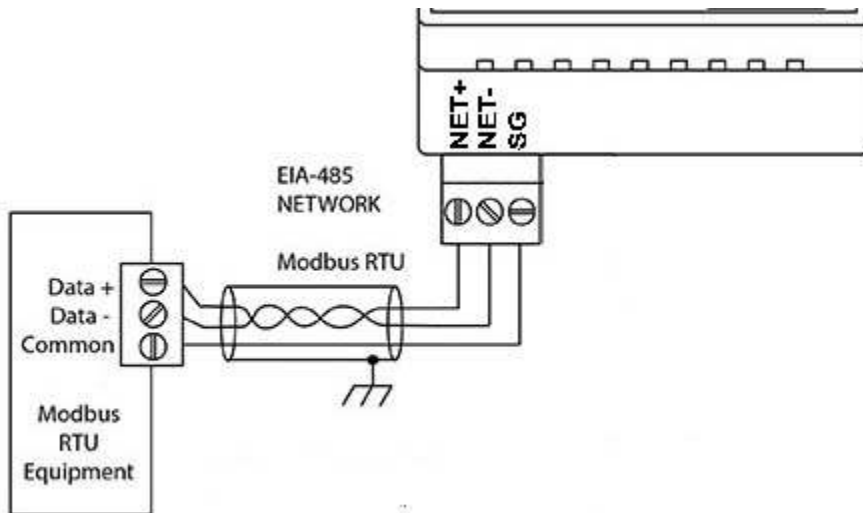
I/O Point Type	Wiring Guide	Additional Information
<p>Discrete Output: Form A Relay</p>		<p>DO outputs 1-2 are Form A dry contact relays rated for 2A @ 120VAC (resistive). Snubbers should be used with inductive loads. The relays are also rated for 2A @ 30VDC. Note: Relays are rated higher, but UL listing is for 2A.</p>

Connection of Power and Communications

Power and communications should be connected as indicated below.

I/O Point Type	Wiring Guide	Additional Information
Power	Connect AC or +DC power to Power terminal. Connect common or -DC power to GND terminal. GND is common to all terminals labeled COM.	Nominal power consumption is 2.4 watts, or 0.1A @ 24VDC, with all relays on.
Communications	Connect Modbus RTU RS-485 data lines to NET+/- terminals. The RS-485 port is electrically isolated from all other terminals including power. Therefore, the SG must be connected to the ground reference for the RS-485 signal in order to communicate.	Communication signals comply with EIA-485 standard.
Wiring Terminals	 <p>The diagram shows a terminal block with the following connections:</p> <ul style="list-style-type: none"> RS-485 Modbus RTU: NET+ (top), NET- (middle), SG (bottom). Sensor Inputs: A/UI1, A/UI2, A/UI3, A/UI4, A/UI5, A/UI6. Power Input: POWER, GND. Relay Outputs: COM, DO 1, DO 2, GND. Sensor Inputs: A/UI7, A/UI8, A/UI9, GND, A/UI10, A/UI11, A/UI12, GND. 	<p>Screw terminals ratings are substantially in excess of any I/O point ratings.</p> <p>Screw terminals are pluggable. They unplug from the unit in 3, 8, and 12-position blocks.</p>

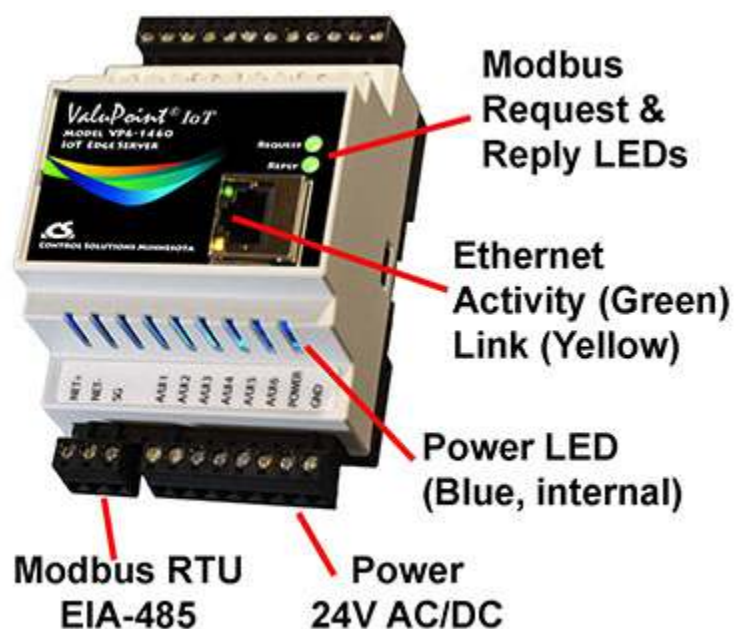
The RS-485 should be wired as illustrated below.



A.2 Front Panel LED Indicators

VP6-1460 Power-up LED behavior: On power up, the Reply LED will remain on solid red for about 20 seconds, then the Request and Reply LEDs will do a "lamp test"

where Request is yellow and Reply is Red simultaneously for about 1 second, and then both Request and Reply turn green simultaneously for about 1 second. The LEDs will then begin to operate according to their normal functionality.



VP6-1460 Request and Reply LEDs reflect Modbus RTU traffic, and the Ethernet activity LED will indicate network traffic in general. If Modbus RTU is not being used at all, then the Request and Reply LEDs will indicate TCP traffic. If Modbus RTU is in use, then the Request and Reply LEDs will indicate Modbus RTU traffic while the Ethernet LEDs will be the only indication of TCP traffic.

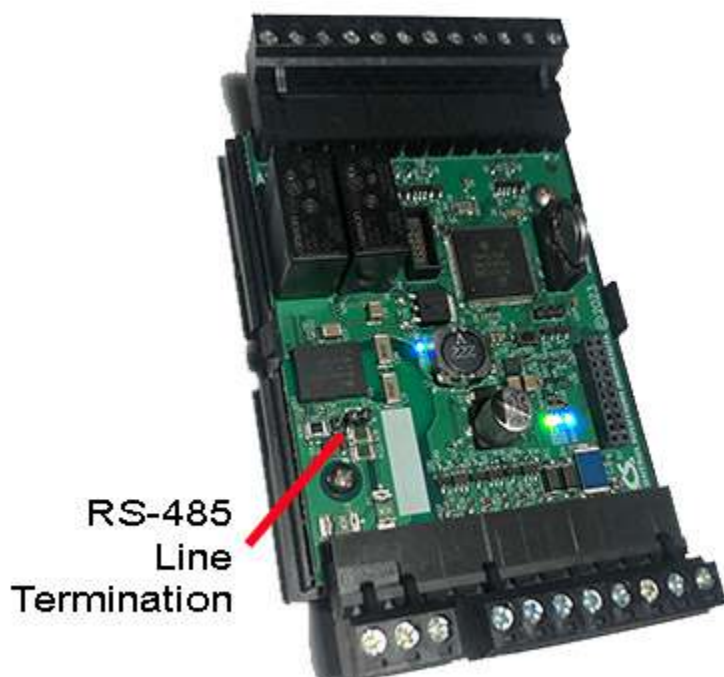
VP6-1460 LEDs indicate as follows (LEDs are bi-color):

REQUEST	Flashes yellow each time a request is sent when operating as Modbus Master, or each time a request is received when operating as Modbus Slave.
REPLY	Operating as Modbus Master, flashes green each time a good response is received, or red when an error code is received, the request times out, or there is a flaw in the response such as CRC error. Operating as Modbus Slave, flashes green each time a good response is sent, or red if an exception code is sent (meaning the received request resulted in an error).
Ethernet Activity	Green LED is on solid during portions of the boot-up process, and then flashes briefly when Ethernet network traffic is detected.
Ethernet Link	Yellow LED indicates an Ethernet link is present. This indicator will light if a link is present regardless of processor or network activity. If not lit, check network wiring.

Status	Blue LED (internal) on any time power is present and internal power supply is functioning. In addition, a green "heartbeat" LED on the base board will blink about once every 2 seconds.
--------	--

A.3 RS-485 Line Termination

RS-485 line termination jumper is located as indicated in the photo. The line termination is "on" when the jumper block is aligned with the screened white block on the circuit board. Termination is required when the VP6-1460 is the last device on the daisy chain.



A.4 Soft Configuration Reset

Soft reset should be used to remove all configuration information any time you do have the ability to connect to the ValuPoint's web user interface. The "Clear Configuration" action is described in Section 3.1.5. Using the forced hard reset should only be used as a last resort if you are unable to connect to the ValuPoint because the SSL certificates are invalid for a secure connection or you are unable to recover the lost IP address.

A.5 Discovering Lost IP Address

You can use Wireshark to discover a lost IP address if the ValuPoint is still functional. Connect the ValuPoint directly to your PC running Wireshark using a cross-over cable (or standard CAT5 cable if your PC supports auto-MDX). With Wireshark running, power up the ValuPoint.

Upon power up, VP6-1460 will ping its own IP address one or more times. This is part

of its duplicate address resolution mechanism. If it finds another device with its own IP address, it will set its own IP address to a default pseudo-random address generally starting with 192.

Wait until you are certain ValuPoint has booted up, or wait 2-3 minutes to be sure if you don't recognize the bootup LED sequence. Now look for the ARP packets and note what IP address they came from. This is your device. (To make sure it is your device, connect only the ValuPoint to your PC while doing this exercise.)

Your device will have a MAC address that starts with 00:40:9D, also labeled with a source that starts with "Digiboar_". This label comes from the fact that the server modules used on Control Solutions IP products are made by Digi International, previously known as "Digiboard".

There will usually be one or more "pings" or ARP packets to the device's own IP address, and one last ping to its own address plus one. In the illustration here, the ValuPoint is located at 192.168.1.42.

(Untitled) - Wireshark

File Edit View Go Capture Analyze Statistics Help

Filter: Expression... Clear Apply

No. -	Time	Source	Destination	Protocol	Info
1	0.000000	Ibm_5e:b7:30	Broadcast	ARP	Gratuitous ARP for 192.168.1.25 (Request)
2	0.999898	Ibm_5e:b7:30	Broadcast	ARP	Gratuitous ARP for 192.168.1.25 (Request)
3	2.001429	Ibm_5e:b7:30	Broadcast	ARP	Gratuitous ARP for 192.168.1.25 (Request)
4	3.031837	Ibm_5e:b7:30	Broadcast	ARP	who has 192.168.1.1? Tell 192.168.1.25
5	3.032390	192.168.1.25	239.255.255.250	IGMP	V2 Membership Report / Join group 239.255.2
6	4.004269	192.168.1.25	239.255.255.250	IGMP	V2 Membership Report / Join group 239.255.2
7	4.024360	Ibm_5e:b7:30	Broadcast	ARP	who has 192.168.1.1? Tell 192.168.1.25
8	5.005791	192.168.1.25	239.255.255.250	IGMP	V2 Membership Report / Join group 239.255.2
9	5.025819	Ibm_5e:b7:30	Broadcast	ARP	who has 192.168.1.1? Tell 192.168.1.25
10	5.034508	Ibm_5e:b7:30	Broadcast	ARP	who has 192.168.1.1? Tell 192.168.1.25
11	39.073317	Digiboar_2e:de:3f	Broadcast	ARP	Gratuitous ARP for 192.168.1.42 (Request)
12	39.289914	Digiboar_2e:de:3f	Broadcast	ARP	who has 192.168.1.43? Tell 192.168.1.42
13	43.994394	192.168.1.42	224.0.0.128	IGMP	V2 Membership Report / Join group 224.0.0.1

Frame 12 (60 bytes on wire, 60 bytes captured)

Ethernet II, Src: Digiboar_2e:de:3f (00:40:9d:2e:de:3f), Dst: Broadcast (ff:ff:ff:ff:ff:ff)

Address Resolution Protocol (request)

Hardware type: Ethernet (0x0001)
 Protocol type: IP (0x0800)
 Hardware size: 6
 Protocol size: 4
 Opcode: request (0x0001)
 Sender MAC address: Digiboar_2e:de:3f (00:40:9d:2e:de:3f)
 Sender IP address: 192.168.1.42 (192.168.1.42)
 Target MAC address: 00:00:00:00:00:00 (00:00:00:00:00:00)
 Target IP address: 192.168.1.43 (192.168.1.43)

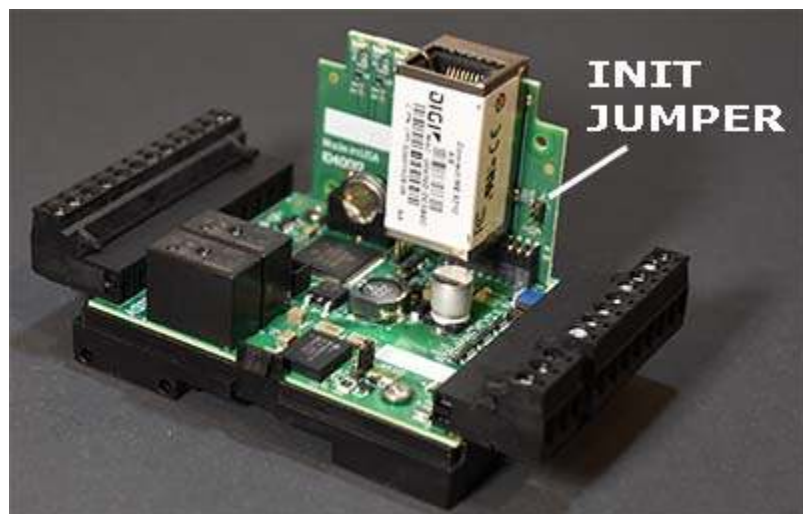
0000 ff ff ff ff ff ff 00 40 9d 2e de 3f 08 06 00 01
 0010 08 00 06 04 00 01 00 40 9d 2e de 3f c0 a8 01 2a
 0020 00 00 00 00 00 00 c0 a8 01 2b a4 d6 d5 d9 d1 d1
 0030 00 00 00 00 00 00 00 00 00 00 00 00

Frame (frame), 60 bytes Packets: 13 Displayed: 13 Marked: 0 Dropped: 0 Profile: Default

A.6 Forced Hard Configuration Reset

IMPORTANT: Before considering the forced hard reset, be sure you have considered soft configuration reset, or discovering lost IP address if applicable.

The "Init" jumper inside the VP6-1460 serves two purposes, and what it does depends on whether you apply the jumper before or after the ValuPoint boots up.



Hard Configuration Reset:

Installing the jumper after bootup causes the ValuPoint to do a hard reset on its configuration memory. The IPv4 address will be reset to 10.0.0.101. The root password will be reset to the original default password. After clearing all configuration, the ValuPoint will begin to flash its LEDs about once a second indefinitely until you remove the jumper, and then it will automatically restart.

Once you have regained access to the device, go to the File Manager page, execute the Clear All configuration action, then select the file named as "Boot configuration" and execute the Save XML Config File action to wipe out any configuration normally saved in the XML configuration file.

Note: The forced hard reset will restore HTTP web access and disable HTTPS web access. The forced hard reset will also restore FTP access to allow FTP firmware uploads if needed.

Note: The hard reset of configuration also means all of your resource allocations are reset to original factory defaults. If you want resource allocations that are different, you will need to repeat the allocation setup as described in Section 3.4.

Firmware Update Recovery:

Installing this jumper prior to power-up causes the server to go into TFTP firmware update mode. Normally you would perform a firmware update by simply uploading a new image.bin file (provided by Control Solutions tech support) using the ValuPoint's internal FTP server and a command line FTP session on your PC (Linux or Windows command line). Detailed instructions are included in the zip file that also contains the applicable image.bin file.

Should the FTP upload fail for some reason, then you need to resort to the TFTP

upload method as the fallback method. Full details on how to go about this can be found under the topic "Restoring a corrupt application image" at <https://info.csimn.com>.

Additional maintenance page:

Go to [http\(s\)://10.0.0.101/html/pgRestoreAddr.html](http(s)://10.0.0.101/html/pgRestoreAddr.html) to find the following page (substituting your IP address). It serves two purposes as noted below, which ideally you will never have a use for.



ValuPoint IoT
MODEL VP6-1460
IOT EDGE SERVER

CONTROL SOLUTIONS MINNESOTA

Valid MAC Address Restore

Reformat Flash file system Wipe

File System Wipe:

On rare occasion, the Flash file system has been observed to get corrupted as a result of losing power while a write operation was in progress. This is most effectively confirmed by opening a command prompt FTP session (Windows 10 PowerShell) to try to view the files in the Flash file system. If FTP fails to show any files, in addition to other problems saving or loading files, it may be that the file system has gotten corrupted. If this happens, go to the page pictured above, and enter the Reformat key, then click Wipe, and then power cycle the device (or restart from the File Manager page). The reformat key is 55AAAA55. Simply type that into the window next to the Wipe button.

MAC Address Restore:

In the event the MAC address has been reset due to NVRAM checksum failure, this page will permit restoring the MAC address to its original address as printed on the component label internal to this device, or on the default password label found on the outside or on external documentation included with the device.

If the MAC address is deemed to be valid, the window will be labeled "Valid MAC Address" and you will not be allowed to change it. If the MAC address is deemed to be invalid, the window will be labeled "Restore MAC Address" and you should then enter the correct MAC address and click Restore. A restart is then needed.

A.7 Firmware Update Notes

The most up to date firmware is shipped with all new devices. This isn't like a new

laptop where you spent the first half a day updating software on a computer you thought was brand new. If you believe you have discovered an issue that you believe a firmware update might fix, contact technical support first to confirm whether that is the case, and then to get a login to the firmware update support site.

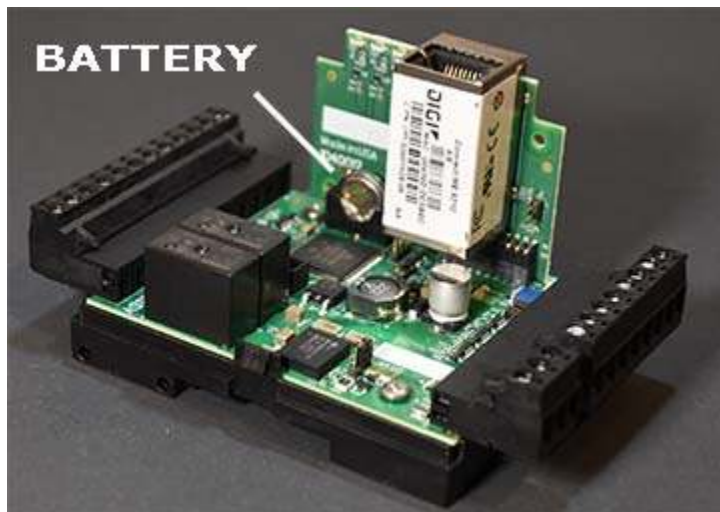
The brute force approach to updating firmware using TFTP as noted in the section above is always available, but the more graceful approach is to use FTP to upload the new image.bin file. There is one minor problem: The upload wants to buffer the entire file in RAM while it proceeds to reprogram the Flash memory. **If the memory utilization indicated on the Resources page in your device is above about 30%, the FTP upload will fail, and thus the firmware update will not take place.**

You have two choices: (1) Use the TFTP approach, or (2) Temporarily reconfigure your ValuPoint to use a minimum of resources to free up space to temporarily buffer the image.bin file upload.

More detailed instructions for the FTP upload are included in the zip file you will download to obtain the firmware update. Instructions for the TFTP upload are available in our knowledgebase at <https://info.csimn.com>.

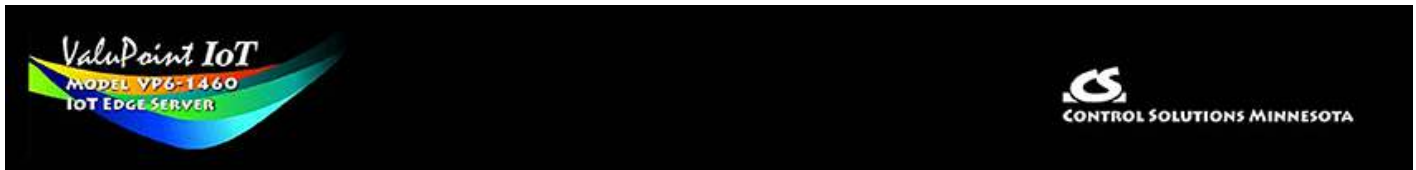
A.8 Battery Replacement

No action is required of the user to activate the battery that backs up the real time clock. The battery should have a 10-year life in normal operation with "normal" meaning that the device is normally continuously powered for use with only intermittent dependency on the battery backup.



Replace battery with BR1225A only. Use of another battery may present a risk of fire or explosion. Replace with battery polarity as marked on the circuit board. Reverse polarity of the battery will not damage the board, but the clock will not be backed up if the battery is reversed.

CAUTION: The lithium battery contained in this device may explode if mistreated. DO NOT recharge, disassemble, or dispose of in fire.



Appendix B Modbus CSV Import Files

B.1 Modbus RTU Master Read/Write Maps

The CSV file for configuring Modbus TCP client read and write maps should contain a single header line with the labels indicated below, and content as applicable.

Header Line Label	Notes	Description of Use
RW	-	Enter 'R' to Read from a remote device, or 'W' to Write to a remote device.
TYPE	-	Use this column to specify remote registers by type (see Reference B)
REG	-	Use this column in conjunction with Type to specify remote register numbers of the selected type. Note that register numbers are 1-indexed, meaning raw address 0 should be entered as register #1.
FORMAT	-	Specify the format of the remote register to be read or written (see Reference C)
SLAVE	-	Provide the slave address, ID, or unit number, of the Modbus RTU slave to be polled.
SWAP	-	Any data item that occupies more than one Modbus register, e.g. 32-bit or Float, needs to have the register order defined since this is not standardized by Modbus protocol. The ValuPoints default to the high order register first. If the remote Modbus slave has its registers ordered with low order first, then select 'T' (True) to "swap" the register order (or 'F' to keep the default order).
SCALE	-	Data is multiplied by this scale factor after read from a remote device or before being written to a remote device.
OFFSET	-	This offset is added to the data value after read from a remote device or before being written to a remote device.
POLL	-	Specify a periodic poll time in seconds (fractions of sections are recognized).
LOCALREG	-	Specify a local Modbus register number where data read from a remote device will be placed, or where data written to a remote device will be taken from.
MASK	-	When READING: If a bit mask is entered (in hexadecimal), and the remote register type is signed or unsigned integer, the mask will be bit-wise logical AND-ed with the data, and

		<p>the retained bits will be right justified in the result.</p> <p>When WRITING: If a bit mask is entered, and the remote register type is signed or unsigned, the mask will be bit-wise logical AND-ed with the data. The mask is right justified, then AND-ed with the data. The result is then left shifted back to the original position of the mask. In other words, the least significant bits of the original data will be stuffed at the position marked by the mask.</p>
DEFAULT	-	<p>When READING: The default value will be stored into the local object/register after the given number of read failures if the fail count (MAXFAIL) is non-zero.</p> <p>When WRITING: The default value will be stored into the local object/register if POR is set to True, or if the amount of time specified by TIMEOUT is exceeded without an update to the local object by a remote client.</p>
MAXFAIL	1	If non-zero, sets the maximum number of times that a read attempt may fail before the default value will be placed in the local object/register. Setting the count to zero will disable the default, and the object/register will retain the most recent value obtained.
FILL	2	When WRITING: The bit fill will be logically OR-ed into the result, but only if the mask was nonzero and was used. Both mask and fill are entered in hexadecimal.
MAXQUIET	2	If using 'send on delta', to guarantee that the remote device will be written at least occasionally even if the data does not change, enter a maximum quiet time (in seconds).
MINQUIET	2	If using 'send on delta', and the delta increment is small, the result can be a large amount of network traffic. To limit network traffic, provide a MINQUIET time (in seconds) that must elapse between transmission of changed values.
DELTA	2	The local object/register data may be written to the remote device periodically, or when the local value changes, or both. To send upon change (send on delta), provide a DELTA value as the amount by which the local object/register must change before being written to the remote device. Leave blank if send on delta should not be used.

The minimum required header line for Modbus RTU must include RW, LOCALREG, TYPE, REG, FORMAT, and SLAVE. All other columns are optional.

This is an example of a minimum CSV file as it would appear in a spread sheet program:

	A	B	C	D	E	F	G	H	I	J
1	RW	TYPE	REG	FORMAT	SLAVE	LOCALREG				
2	R	HOLD	40	S16	22	1				
3	R	HOLD	42	U32	22	2				
4	R	HOLD	44	U32	22	3				
5	R	HOLD	46	U32	22	4				
6	R	HOLD	48	U16	22	5				
7	R	HOLD	50	U16	22	6				
8	R	HOLD	52	U16	22	7				
9	R	HOLD	54	S16	22	8				
10	R	HOLD	56	S32	22	9				
11	R	HOLD	58	S32	22	10				
12	R	HOLD	60	S16	22	11				

This is an example of how the minimum CSV file looks as just plain text:

```

rtu.csv - Notepad
File Edit Format View Help
RW,TYPE,REG,FORMAT,SLAVE,LOCALREG
R,HOLD,40,S16,22,1
R,HOLD,42,U32,22,2
R,HOLD,44,U32,22,3
R,HOLD,46,U32,22,4
R,HOLD,48,U16,22,5
R,HOLD,50,U16,22,6
R,HOLD,52,U16,22,7
R,HOLD,54,S16,22,8
R,HOLD,56,S32,22,9
R,HOLD,58,S32,22,10
R,HOLD,60,S16,22,11
R,HOLD,62,U32,22,12
R,HOLD,64,U32,22,13
R,HOLD,66,U32,22,14
    
```

B.2 Modbus TCP Client Read/Write Maps

The CSV file for configuring Modbus TCP client read and write maps should contain a single header line with the labels indicated below, and content as applicable.

The only difference between TCP and RTU formats is DEVNUM and UNIT in TCP, versus just SLAVE in RTU. Everything else is identical.

Header Line	Notes	Description of Use
-------------	-------	--------------------

Label		
RW	-	Enter 'R' to Read from a remote device, or 'W' to Write to a remote device.
TYPE	-	Use this column to specify remote registers by type (see Reference B)
REG	-	Use this column in conjunction with Type to specify remote register numbers of the selected type. Note that register numbers are 1-indexed, meaning raw address 0 should be entered as register #1.
FORMAT	-	Specify the format of the remote register to be read or written (see Reference C)
DEVNUM	-	Specify the device number where the remote register is to be found. This number is used to look up a device in the Modbus TCP Client Device table which contains the device's IP address, etc.
UNIT	-	Unit number is optional, and may be 1 to 247. (BB2-6010, BB2-7010 only)
SCALE	-	Data is multiplied by this scale factor after read from a remote device or before being written to a remote device.
OFFSET	-	This offset is added to the data value after read from a remote device or before being written to a remote device.
POLL	-	Specify a periodic poll time in seconds (fractions of sections are recognized).
LOCALREG	-	Specify a local Modbus register number where data read from a remote device will be placed, or where data written to a remote device will be taken from.
MASK	-	<p>When READING: If a bit mask is entered (in hexadecimal), and the remote register type is signed or unsigned integer, the mask will be bit-wise logical AND-ed with the data, and the retained bits will be right justified in the result.</p> <p>When WRITING: If a bit mask is entered, and the remote register type is signed or unsigned, the mask will be bit-wise logical AND-ed with the data. The mask is right justified, then AND-ed with the data. The result is then left shifted back to the original position of the mask. In other words, the least significant bits of the original data will be stuffed at the position marked by the mask.</p>
DEFAULT	-	<p>When READING: The default value will be stored into the local object/register after the given number of read failures if the fail count (MAXFAIL) is non-zero.</p> <p>When WRITING: The default value will be stored into the local object/register if POR is set to True, or if the amount of time specified by TIMEOUT is exceeded without an update to the local object by a remote client.</p>

MAXFAIL	1	If non-zero, sets the maximum number of times that a read attempt may fail before the default value will be placed in the local object/register. Setting the count to zero will disable the default, and the object/register will retain the most recent value obtained.
FILL	2	When WRITING: The bit fill will be logically OR-ed into the result, but only if the mask was nonzero and was used. Both mask and fill are entered in hexadecimal.
MAXQUIET	2	If using 'send on delta', to guarantee that the remote device will be written at least occasionally even if the data does not change, enter a maximum quiet time (in seconds).
MINQUIET	2	If using 'send on delta', and the delta increment is small, the result can be a large amount of network traffic. To limit network traffic, provide a MINQUIET time (in seconds) that must elapse between transmission of changed values.
DELTA	2	The local object/register data may be written to the remote device periodically, or when the local value changes, or both. To send upon change (send on delta), provide a DELTA value as the amount by which the local object/register must change before being written to the remote device. Leave blank if send on delta should not be used.

The minimum required header line for Modbus TCP must include RW, LOCALREG, TYPE, REG, FORMAT, and DEVNUM. All other columns are optional.

B.3 Register Types

The content of the TYPE column should contain one of the following CSV Labels:

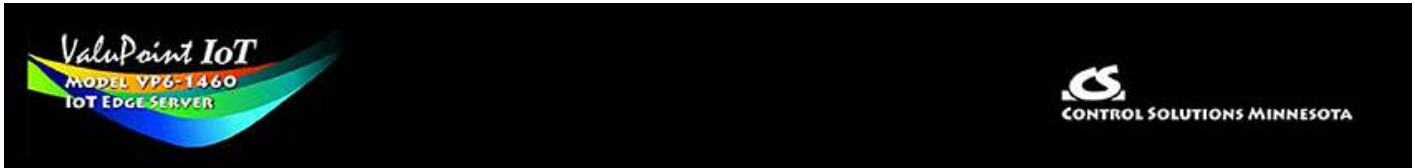
CSV Label	Modbus Register Type	Function Code for Read	Function Code for Write
COIL	Coil (1 bit)	1	5 or 15
DISC	Discrete Input (1 bit)	2	n/a
INPUT	Input Register (16 bits)	4	n/a
HOLD	Holding Register (16 bits)	3	6 or 16

B.4 Register Formats

The content of the FORMAT column should contain one of the following CSV Labels:

CSV Label	Modbus Register Data Format	Occupies # Registers
S16	Signed 16-bit Integer	1
U16	Unsigned 16-bit Integer	1
S32	Signed 32-bit Integer	2
U32	Unsigned 32-bit Integer	2
FP	Floating Point, IEEE 754 32-bit	2

BIT	Bit (only used for Coil or Discrete Input registers)	-
MOD102	Mod-10, 2-register	2
MOD103	Mod-10, 3-register	3
MOD104	Mod-10, 4-register	4
S64	Signed 64-bit Integer	4



Appendix C Trouble Shooting

C.1 Modbus RTU Trouble Shooting

You will find message and error counters listed on the Error Counts page under RTU Data. If the ValuPoint is configured as Modbus master, then the Error Counts page will list counts by slave address. If the ValuPoint is configured as Modbus slave, then errors show up on the first line (Unit # 1) regardless of what address the ValuPoint is configured to be.

The Errors: Read Maps and Errors: Write Maps pages will tell you exactly which maps are getting errors when the ValuPoint is configured as Modbus Master.

The most frequent problem is "no response" or timeout. This means the master and slave are not connecting for any of several possible reasons: (a) There is a wiring problem; (b) Port parameters are not configured the same (baud rate, etc); (c) Master's timeout setting is too short.

When it comes to wiring, remember that RS-485 is NOT truly a 2-wire interface as it is commonly referred to. Refer to the RS-485 FAQ under Support at csimn.com if you have questions or concerns about wiring.

If you are getting CRC errors, that is almost always a wiring problem, but can be a port problem such as mismatched parity setting. A CRC error will not be caused by incorrect configuration of a Read Map or Write Map.

If you are getting exception errors, that is somewhat good news - it means that at least you are successfully communicating. An exception error most often means the master is asking the slave for a register that the slave does not have. If the ValuPoint is configured as Modbus master, this means the Read Map or Write Map is not configured correctly.

C.2 Modbus TCP Trouble Shooting

You will find message and error counters listed on the Error Counts page under TCP Data for Modbus client activity. Counts will be listed by device number for those devices found on the TCP Setup Devices page.

The Errors: Read Maps and Errors: Write Maps pages will tell you exactly which maps are getting errors when the ValuPoint is operating as Modbus TCP client (master).

The most frequent problem is "no response" or timeout. The most common cause of this problem for Modbus TCP is a network configuration problem, such as incorrect IP address or IP address that cannot be reached as configured. The problem sometimes

lies outside the ValuPoint and may require consulting with the IT personnel responsible for the network if on a large network.

If you are getting exception errors, that is somewhat good news - it means that at least you are successfully communicating. An exception error most often means the master is asking the slave for a register that the slave does not have. If the ValuPoint is configured as Modbus master, this means the Read Map or Write Map is not configured correctly.

C.3 AWS Trouble Shooting

You can verify the content of Publish and Subscribe messages using the IoT Cloud :: Thing Status :: Test page once you have established a successful connection to the AWS server. The status of the connection will be indicated on the Thing Status :: Connection page.

One hidden factor that can prevent making a connection is lack of a valid payment method in your AWS account. If your credit card has expired, the ValuPoint won't tell you this - it will only tell you that AWS is refusing to connect.

When first setting up a new Thing, two of the more hidden factors that can interfere with connecting are lack of a DNS server (set up on Network page) or lack of an NTP server (also set up on the Network page). If NTP is not providing correct local time and date for the ValuPoint, then the secure connection will fail. Valid timestamp is part of the SSL handshake protocol.

Making sure your certificates match the Thing and are assigned to the correct usage on the Thing Files page will be required. Making sure the host name and thing name on the Thing ID page are correct will also be required.

There are a number of reasons why the ValuPoint will not connect to the AWS server, but the above items cover all of the most common reasons. The reason will generally be indicated with a verbose description on the Connection page.

C.4 Modbus Reference Information

Modbus Register Types

The types of registers referenced in Modbus devices include the following:

- Coil (Discrete Output)
- Discrete Input
- Input Register
- Holding Register

Whether a particular device includes all of these register types is up to the manufacturer. It is very common to find all I/O mapped to holding registers only. Coils are 1-bit registers, are used to control discrete outputs, and may be read or written. Discrete Inputs are 1-bit registers used as inputs, and may only be read. Input registers are 16-bit registers used for input, and may only be read. Holding registers are the most universal 16-bit register, may be read or written, and may be used for a

variety of things including inputs, outputs, configuration data, or any requirement for "holding" data.

Modbus Function Codes

Modbus protocol defines several function codes for accessing Modbus registers. There are four different data blocks defined by Modbus, and the addresses or register numbers in each of those overlap. Therefore, a complete definition of where to find a piece of data requires both the address (or register number) and function code (or register type).

The function codes most commonly recognized by Modbus devices are indicated in the table below. This is only a subset of the codes available - several of the codes have special applications that most often do not apply.

Function Code	Register Type
1	Read Coil
2	Read Discrete Input
3	Read Holding Registers
4	Read Input Registers
5	Write Single Coil
6	Write Single Holding Register
15	Write Multiple Coils
16	Write Multiple Holding Registers

Modbus Exception (error) Codes

When a Modbus slave recognizes a packet, but determines that there is an error in the request, it will return an exception code reply instead of a data reply. The exception reply consists of the slave address or unit number, a copy of the function code with the high bit set, and an exception code. If the function code was 3, for example, the function code in the exception reply will be 0x83. The exception codes will normally be one of the following:

1	Illegal Function	The function code received in the query is not recognized by the slave or is not allowed by the slave.
2	Illegal Data Address	The data address (register number) received in the query is not an allowed address for the slave, i.e., the register does not exist. If multiple registers were requested, at least one was not permitted.
3	Illegal Data Value	The value contained in the query's data field is not acceptable to the slave.

Modicon convention notation for Modbus registers

Modbus was originally developed by Gould-Modicon, which is presently Schneider Electric. The notation originally used by Modicon is still often used today, even though considered obsolete by present Modbus standards. The advantage in using the

Modicon notation is that two pieces of information are included in a single number: (a) The register type; (b) The register number. A register number offset defines the type.

The types of registers referenced in Modbus devices, and supported by ValuPoints, include the following:

- Coil (Discrete Output)
- Discrete Input
- Input Register
- Holding Register

Valid address ranges as originally defined for Modbus were 0 to 9999 for each of the above register types. Valid ranges allowed in the current specification are 0 to 65,535. The address range applies to each type of register, and one needs to look at the function code in the Modbus message packet to determine what register type is being referenced. The Modicon convention uses the first digit of a register reference to identify the register type.

Register types and reference ranges recognized by ValuPoints are as follows:

0x = Coil = 00001-09999
1x = Discrete Input = 10001-19999
3x = Input Register = 30001-39999
4x = Holding Register = 40001-49999

Translating references to addresses, reference 40001 selects the holding register at address 0000, most often referred to as holding register number 1. The reference 40001 will appear in documentation using Modicon notation, but ValuPoints require specifying "holding register" and entering that register number as just "1".

On occasion, it was necessary to access more than 10,000 of a register type using Modicon notation. Based on the original convention, there is another defacto standard that looks very similar. Additional register types and reference ranges recognized by ValuPoints are as follows:

0x = Coil = 000001-065535
1x = Discrete Input = 100001-165535
3x = Input Register = 300001-365535
4x = Holding Register = 400001-465535

If registers are 16-bits, how does one read Floating Point or 32-bit data?

Modbus protocol defines a holding register as 16 bits wide; however, there is a widely used defacto standard for reading and writing data wider than 16 bits. The most common are IEEE 754 floating point, and 32-bit integer. The convention may also be extended to double precision floating point and 64-bit integer data.

The wide data simply consists of two consecutive "registers" treated as a single wide register. Floating point in 32-bit IEEE 754 standard, and 32-bit integer data, are widely used. Although the convention of register pairs is widely recognized, agreement on whether the high order or low order register should come first is not standardized. For this reason, many devices, including all Control Solutions devices, support register

"swapping". This means you simply check the "swapped" option (aka "High reg first" in some devices) if the other device treats wide data in the opposite order relative to Control Solutions default order.

Control Solutions Modbus products all default to placing the high order register first, or in the lower numbered register. This is known as "big endian", and is consistent with Modbus protocol which is by definition big endian.

What does notation like 40001:7 mean?

This is a commonly used notation for referencing individual bits in a register. This particular example, 40001:7, references (Modicon) register 40001, bit 7. Bits are generally numbered starting at bit 0, which is the least significant or right most bit in the field of 16 bits found in a Modbus register.

How do I read individual bits in a register?

Documentation tends to be slightly different for every Modbus device. But if your device packs multiple bits into a single holding register, the documentation will note up to 16 different items found at the same register number or address. The bits may be identified with "Bn" or "Dn" or just "bit n". Most of the time, the least significant bit will be called bit 0 and the most significant will be bit 15. It is possible you could find reference to bit 1 through bit 16, in which case just subtract one from the number to reference the table below.

You cannot read just one bit from a holding register. There is no way to do that - Modbus protocol simply does not provide that function. You must read all 16 bits, and then test the individual bit you are interested in for true or false (1 or 0). ValuPoints provide an automatic way of doing that by including a "mask" in each register map or rule. Each time the register is read, the mask will be logically AND-ed with the data from the register, and the result will be right justified to yield a 1 or 0 based on whether the selected bit was 1 or 0. ValuPoints provide optimization when successive read maps or rules are selecting different bits from the same register. The Modbus register will be read from the slave once, and the 16-bit data will be shared with successive maps or rules, with each map or rule selecting its bit of interest.

The bit mask shown in the expanded form of the ValuPoint RTU read map is a 4 digit hexadecimal (16 bit) value used to mask out one or more bits in a register. The selected bits will be right justified, so a single bit regardless of where positioned in the source register will be stored locally as 0 or 1. The hex bit mask values would be as follows:

B0/D0/bit 0 mask = 0001
B1/D1/bit 1 mask = 0002
B2/D2/bit 2 mask = 0004
B3/D3/bit 3 mask = 0008
B4/D4/bit 4 mask = 0010
B5/D5/bit 5 mask = 0020
B6/D6/bit 6 mask = 0040
B7/D7/bit 7 mask = 0080

B8/D8/bit 8 mask = 0100
B9/D9/bit 9 mask = 0200
B10/D10/bit 10 mask = 0400
B11/D11/bit 11 mask = 0800
B12/D12/bit 12 mask = 1000
B13/D13/bit 13 mask = 2000
B14/D14/bit 14 mask = 4000
B15/D15/bit 15 mask = 8000

Some Modbus devices also pack two 8-bit values into a single 16-bit register. The two values will typically be documented as "high byte" and "low byte" or simply have "H" and "L" indicated. If you run into this scenario, the masking for bytes is as follows:

High byte mask = FF00
Low byte mask = 00FF

When the mask value in a ValuPoint is more than just one bit, the mask is still logically AND-ed with the data from the Modbus slave, and the entire resulting value is right justified to produce an integer value of less than the original bit width of the original register.

There have been a few instances of documenting packed bits in a 32-bit register. Although Modbus protocol is strictly 16-bit registers, some implementations force you to read pairs of registers. If your device documents 32 packed bits, then you would insert 0000 in front of each mask above, and the remainder of the list would be as follows:

B16/D16/bit 16 mask = 00010000
B17/D17/bit 17 mask = 00020000
B18/D18/bit 18 mask = 00040000
B19/D19/bit 19 mask = 00080000
B20/D20/bit 20 mask = 00100000
B21/D21/bit 21 mask = 00200000
B22/D22/bit 22 mask = 00400000
B23/D23/bit 23 mask = 00800000
B24/D24/bit 24 mask = 01000000
B25/D25/bit 25 mask = 02000000
B26/D26/bit 26 mask = 04000000
B27/D27/bit 27 mask = 08000000
B28/D28/bit 28 mask = 10000000
B29/D29/bit 29 mask = 20000000
B30/D30/bit 30 mask = 40000000
B31/D31/bit 31 mask = 80000000

Deciphering Modbus Documentation

Documentation for Modbus is not well standardized. Actually there is a standard, but not well followed when it comes to documentation. You will have to do one or more of the following to decipher which register a manufacturer is really referring to:

a) Look for the register description, such as holding register, coil, etc. If the

documentation says #1, and tells you they are holding registers, then you have holding register #1. You also have user friendly documentation.

b) Look at the numbers themselves. If you see the first register on the list having a number 40001, that really tells you register #1, and it is a holding register. This form of notation is often referred to as the old Modicon convention.

c) Look for a definition of function codes to be used. If you see a register #1, along with notation telling you to use function codes 3 and 16, that also tells you it is holding register #1.

IMPORTANT: Register 1 is address 0. Read on...

d) Do the numbers in your documentation refer to the register number or address? Register #1 is address zero. If it is not clear whether your documentation refers to register or address, and you are not getting the expected result, try plus or minus one for register number. All Control Solutions products refer to register numbers in configuration software or web pages. However, some manufacturers document their devices showing address, not register numbers. When you have addresses, you must add one when entering that register into configuration software from Control Solutions.

Can I put 2 Valupoints on the same Modbus network?

You can not have more than one Master on a Modbus RTU (RS-485) network. Therefore, if the ValuPoint is to be configured as the Master, you can only have 1 ValuPoint. You cannot use multiple ValuPoints to read more points from the same Modbus slave device.

Multiple ValuPoints configured as slaves can reside on the same Modbus RS-485 network.

If you are using RS-232 devices, you can have only two devices total, regardless of how they are configured. RS-232 is not multi-drop.

How many devices can I have on a Modbus RTU network?

Logically you can address over 250 devices; however, the RS-485 transceivers are not capable of physically driving that many devices. Modbus protocol states that the limit is 32 devices, and most RS-485 transceivers will agree with this. Only if all devices on the network have low load transceivers can you have more than 32 devices.



Appendix D SSL Certificates for Secure Web (HTTPS)

The secure web server (HTTPS) requires SSL certificates in order to establish secure connections. These certificates are for the use of the web server, and are not the same or in any way related to the AWS IoT certificates. The HTTPS certificates are only required if HTTPS is enabled on the Network configuration page in the ValuPoint.

D.1 X.509 Auto-Certificate Generation

The ValuPoint will automatically generate X.509 certificates if no external certificates are found or could not be loaded correctly. These will be generated one time and saved in the Flash file system for subsequent reuse. When the self-generated X.509 certificates are in use, this will be indicated at the bottom of the Network configuration page.



If there is a need to delete the self-generated certificates, you can do so by logging in via FTP. Change directory to /FLASH0, then to .cfg. The two certificate files that were self-generated are ssl.cert and ssl.key.



```

C:\Users\Jim Hogenson\My Documents\bb-mq-61\config files>ftp
ftp> open 192.168.1.120
Connected to 192.168.1.120.
220 NET+OS 7.5.2.2 FTP server ready.
User (192.168.1.120:(none)): root
331 User root OK, send password.
Password:
230 Password OK.
ftp> cd /FLASH0
250 Directory is changed
ftp> dir .cfg
200 PORT command Ok.
150 File Listing Follows in ASCII mode
-rwlrwlrw---- 1 noone      group2 447      Dec 31 1969 ssl.cert
-rwlrwlrw---- 1 noone      group2 465      Dec 31 1969 ssl.key
226 Transfer complete.
ftp> 119 bytes received in 0.11Seconds 1.09Kbytes/sec.
ftp>

```

D.2 External Certificates

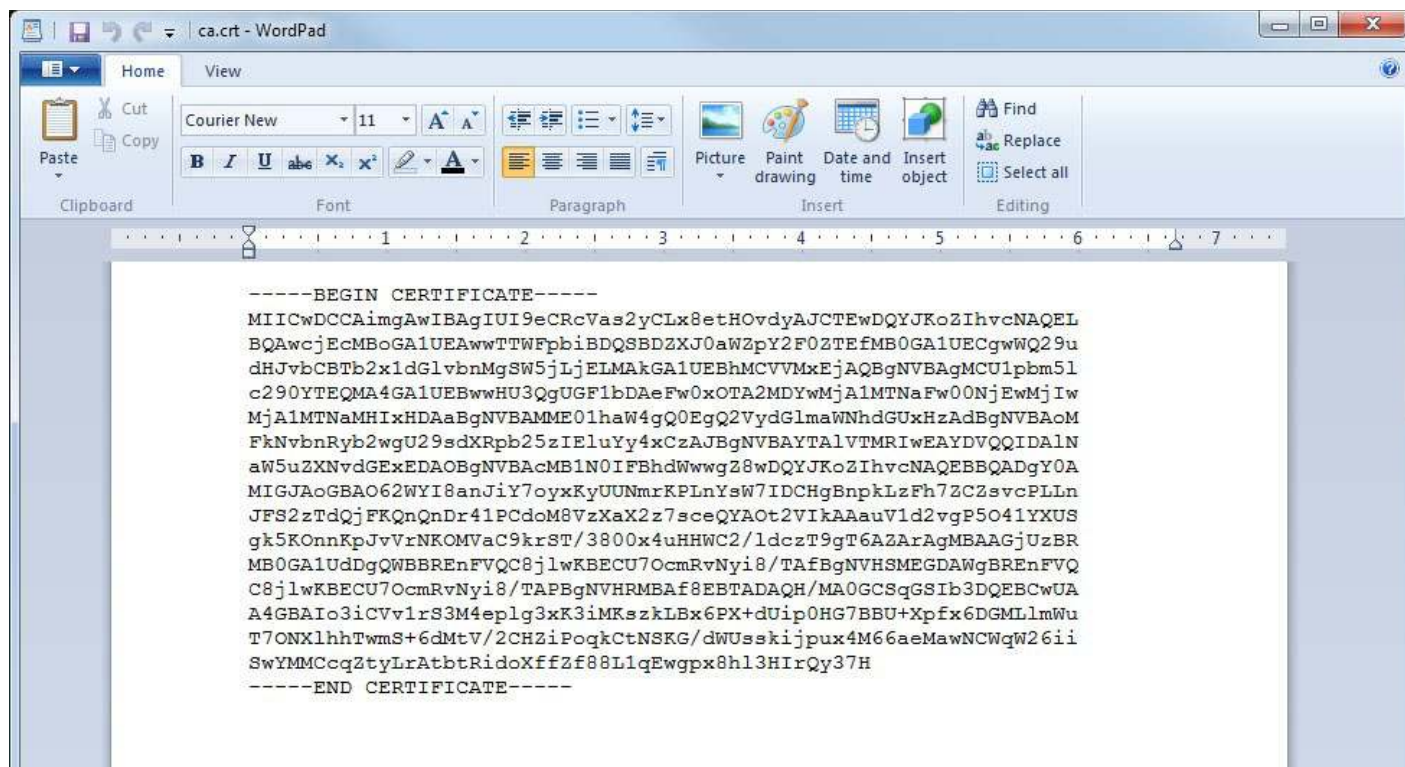
There are three certificates that you must generate and upload to use SSL certificates other than the self-generated X.509 certificates.



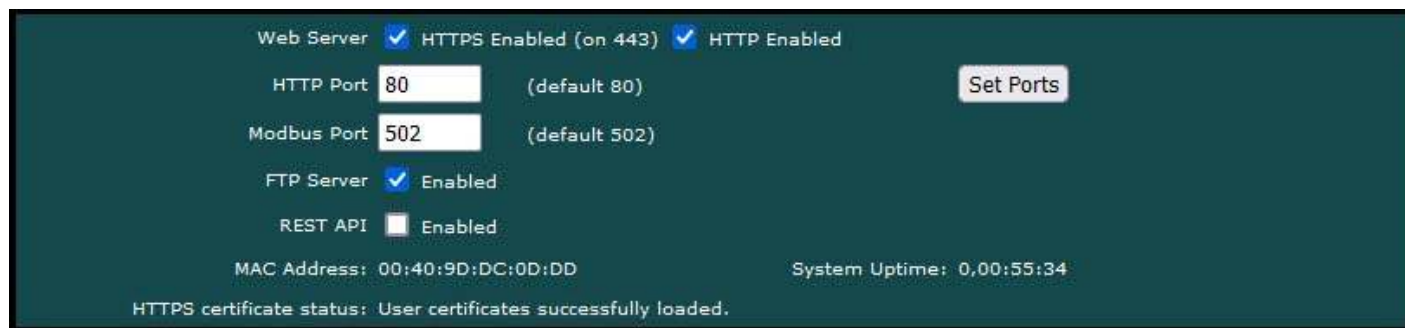
The required certificates are as follows, and must use exactly these names.

ca.crt	CA Root certificate in PEM format
server.crt	Server certificate in PEM format
server.key	Server private key in PEM format

The content of each certificate file will look something like the screen shot below. If you require external certificates for your secure web server, the requirement was likely imposed by your IT department. They should be able to provide the necessary certificates for you. For globally accessed use, the Root CA would come from somebody like GoDaddy or DigiCert (formerly Symantec).



If external certificates were loaded successfully, that will be indicated at the bottom of the Network configuration page.



D.3 Certificate Generation Script (Linux)

The art and science of generating SSL certificates is beyond the scope of this document. An example SSL certificate generation script is provided here as a reference.

The following script, run on a Linux system with OpenSSL installed, will generate the three required SSL certificate files. It will generate a number of intermediate files as well - you don't need to upload them. Replace references to Control Solutions in this script with your own company name.

```
#!/bin/bash
echo hello
# This will create some self signed certs, using one master CA.
#
# these can be the webserver DNS name, or an IP address, however you
access
```

```
# the resource, this needs to match.
if [ -z "$1" ] || [ -z "$2" ]; then
echo 'Usage: gen.sh <server-name> <client-name>'
echo ' <server-name> and <client-name> can be IP addresses'
echo ' or DNS names.'
exit 1
fi
SNAME=$1
CNAME=$2
#
# Bits for strength, 1024, 2048, 4096, etc.. (suggest 2k or 4k for web
servers)
BITS=1024
#
# HASH - Options are sha256, sha512, sha1, md5
HASH="sha256"
SN=`date +%Y%m%d%H%M%S`
#####
# below is the entry for the CRL
# Do not use http://www.csimn.com/crl.pem for production keys and
certificates
# cat <<EOF >> extensions.cnf
# [ extensions_section ]
# crlDistributionPoints = URI:http://www.csimn.com/crl.pem
#
# basicConstraints = CA:FALSE
# keyUsage = nonRepudiation, digitalSignature, keyEncipherment
# subjectAltName = DNS:${SNAME},IP:${SNAME}
# EOF
#####
#####
# first, lets generate some private keys...
openssl genrsa -out server.key ${BITS}
openssl genrsa -out client.key ${BITS}
# ok, and now the MAIN CA
openssl req -x509 -${HASH} -nodes -days 10000 -newkey rsa:${BITS} -keyout
ca.key -out ca.crt -subj "/CN=Main CA Certificate/O=Control Solutions
Inc./C=US/ST=Minnesota/L=St Paul"
#####
#
# Create a CSR for both server and client
# Replace these values with one appropriate for your organization
openssl req -out server.csr -key server.key -new -subj "/CN=${SNAME}/
O=Control Solutions Inc./C=US/ST=Minnesota/L=St Paul"
openssl req -out client.csr -key client.key -new -subj "/CN=${CNAME}/
O=Control Solutions Inc./C=US/ST=Minnesota/L=St Paul"
#
#
#####
# Sign the keys with the CA
openssl x509 -req -days 3650 -in server.csr -CA ca.crt -CAkey ca.key -
set_serial ${SN}01 -out server.crt -${HASH}
openssl x509 -req -days 3650 -in client.csr -CA ca.crt -CAkey ca.key -
```

```
set_serial ${SN}02 -out client.crt -${HASH}
# Create a windows file to import the client keys if needed in this
format
openssl pkcs12 -export -clcerts -in client.crt -inkey client.key -out
client.p12
# Create the client keys as a complete pem file if needed in this format
openssl pkcs12 -in client.p12 -out client-full.pem -clcerts
# mv -f server.key svrkey.pem
# mv -f server.crt svrcert.pem
# mv -f client.key clntkey.pem
# mv -f client.crt clntcert.pem
# cp -f ca.crt cacert.pem
####
# cleanup
# rm -f client.csr server.csr
#DLS 20160420
echo '*****'
echo '* WARNING: Do not use this script to generate production *'
echo '* keys and certificates. This script is for *'
echo '* demonstration purposes only. *'
echo '*****'
```