User Guide

Babel Buster IoT

Model MQ-61
IoT Gateway

Model BB3-6101-MQ
IoT Gateway

Rev. 2.0 – Sept. 2021

NOTE: This version of the user guide applies to firmware revision 3.16.1 and higher. Older firmware does not have many of the features outlined in this version of the user guide. Contact technical support if you have an older MQ-61 that you would like to update.

MQ-61 & BB3-6101-MQ User Guide Contents

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

2 Connecting the IoT Gateway for the First Time
2.1 Overview of IoT Gateway 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 Open Web User Interface

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 Special Features of Local Registers
4.3 Local Register Calculate Rules
4.4 Local Register Copy Rules
4.5 Device Status Reporting

5 Configuring Gateway 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 Gateway 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 Gateway 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 Gateway 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

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 Gateway
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 Gateway to Publish to AWS
14.1 Create and Register a "Thing"
14.2 Create a Policy
14.3 Attach Policy and Thing to SSL Certificates
14.4 Configure IoT Gateway
14.5 Test Publish to Device Shadow

15 Configuring IoT Gateway to Subscribe to AWS
15.1 Configure IoT Gateway
15.2 Use MQTT Test Client to Test Subscription

16 Configuring AWS Simple Notification Service
16.1 Create Topic
16.2 Create Subscription
16.3 Create Action Rule for SNS
16.4 Configure IoT Gateway and Send Text Message

17 Configuring IoT Analytics
17.1 Create Resources
17.2 Build the Pipeline
17.3 Review Data set
17.4 Create Action Rule for IoT Analytics
17.5 Process Data

18 Configuring AWS QuickSight
18.1 Create New Analysis
18.2 Visualize Data

19 Programming with Script Basic
19.1 Creating a Program
19.2 Testing the Program
19.3 Setting the Program to Auto-Run on Startup
19.4 Special Functions
19.4.1 Serial Port Functions
19.4.2 Register Access
19.4.3 Register New-Data Status
19.4.4 Register Access Status
19.4.5 LED Control via Phantom Register Access

19.4.6 Program Watchdog Timer
19.4.7 Error Information Retrieval
19.4.8 IP Address Retrieval
19.4.9 Free Memory
19.5 Example: Program Triggering of Publish to AWS
19.6 Example: Writing to an Alphanumeric Display
19.7 Example: Device and Rule Diagnostics
19.8 Example: Connecting Custom Serial Device to AWS

20 User HTML
20.1 Static HTML
20.2 Dynamic Data Tags in User HTML
20.3 Live JavaScript Gauges

21 REST API
21.1 GET Data from Device
21.2 POST Data to Device

Appendix A Hardware Details
A.1 Wiring
A.2 Front Panel LED Indicators
A.3 RS-485 Line Termination and Bias
A.4 Soft Configuration Reset
A.5 Discovering Lost IP Address
A.6 Forced Hard Configuration Reset
A.7 Firmware Update Notes

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)

Babel Buster IoT
MODEL MQ-61
MODEL BB3-6101-MQ
IOT GATEWAY

CONTROL SOLUTIONS MINNESOTA

1. Introduction

1.1 How to Use This Guide

This user guide provides background information on how the gateway works, and an overview of the configuration process. There are several sections for groups of tabs found in the web interface in the gateway 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 gateway 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 gateway which is generally sufficient for quick reference in setting up the gateway.

NOTE: The screen shots in this user guide were made using a Model MQ-61. The web pages found in the Model BB3-6101-MQ are identical except for the model number displayed at the top of the page.

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.

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.

Babel Buster IoT
MODEL MQ-61
MODEL BB3-6101-MQ
IOT GATEWAY

CONTROL SOLUTIONS MINNESOTA

2. Connecting the IoT Gateway for the First Time

The Babel Buster IoT Gateway model MQ-61 was the first IoT Gateway introduced by Control Solutions. This User Guide was originally written for that model. The BB3-6101-MQ was added later, and is functionally identical. Therefore, this same User Guide applies to both models with the only difference in the web UI being the model number illustrated at the top of the pages.

The difference in hardware is that the BB3-6101-MQ is DIN rail mounted, and also offers the option of an RS-232 serial port in place of the RS-485 port. Other than model number displayed on the web pages, there is no difference in software.

2.1 Overview of IoT Gateway Operation

The Babel Buster IoT Gateway 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 IoT Gateway and the many features of Amazon Web Services.

Are you not a fan of the "cloud"? No problem. The latest release of the Babel Buster IoT Gateway 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 Gateway itself.

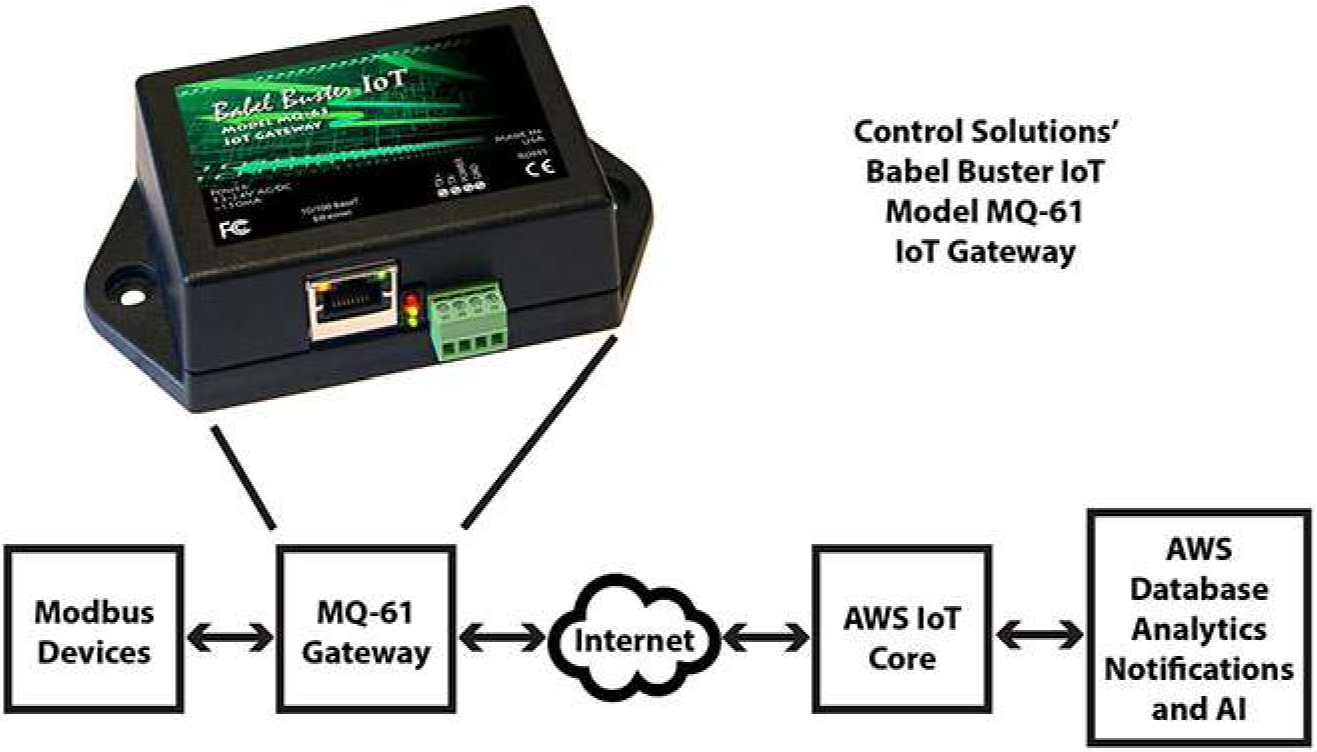
An IoT Device typically has one or more of these functions:

- 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 Babel Buster IoT Gateway.

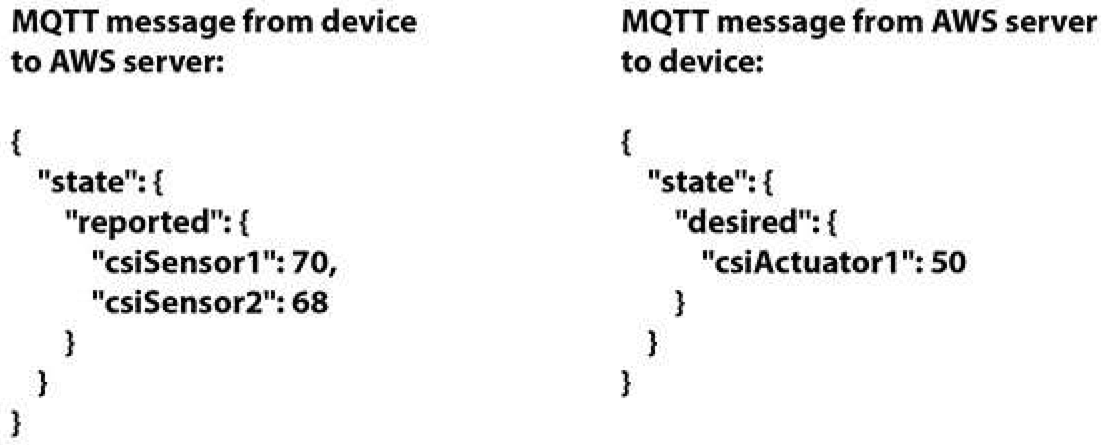
2.1.1 Cloud Based Application

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 Babel Buster IoT Gateway will 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 IoT Gateway will decide if and when to publish that data to the AWS server. You can also configure the IoT Gateway to subscribe to data coming from the AWS server, which you can then write out 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
    }
  }
}
```

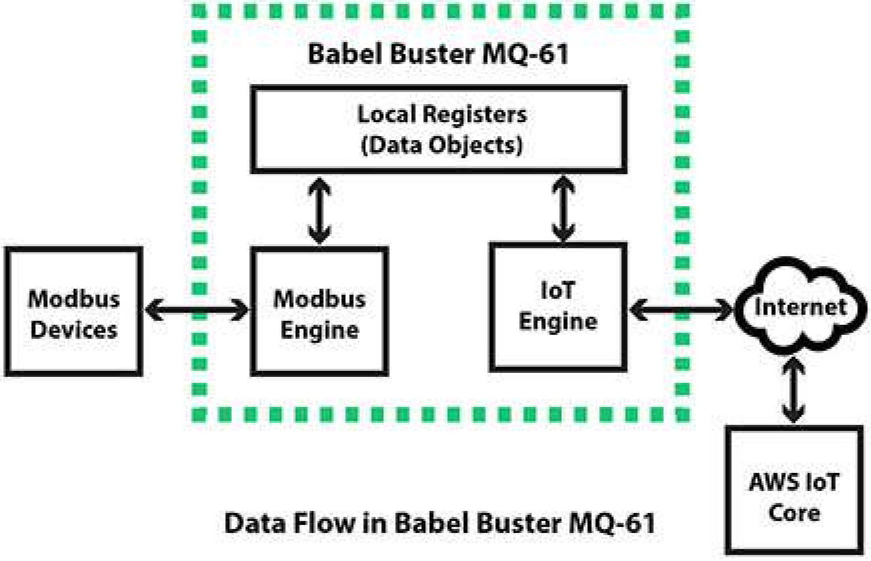
MQTT message from AWS server to device:

```
{
  "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.

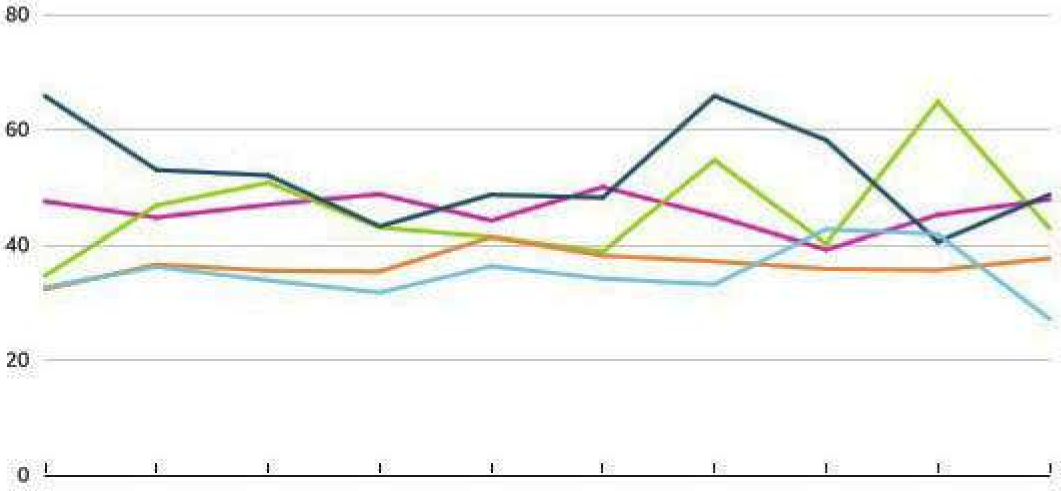
The MQTT "subscribe" action, in controls terms, is most closely associated with actuators. The "subscribe" action would also be associated with control setpoints. You can never force data into a device via MQTT. The device, in our case the IoT Gateway, must subscribe to the source of data effectively 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 IoT Gateway is illustrated below. Data is 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 Modbus 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. These rules are also created by the user and data will be published according to the criteria set up by the user.



Data Flow in Babel Buster MQ-61

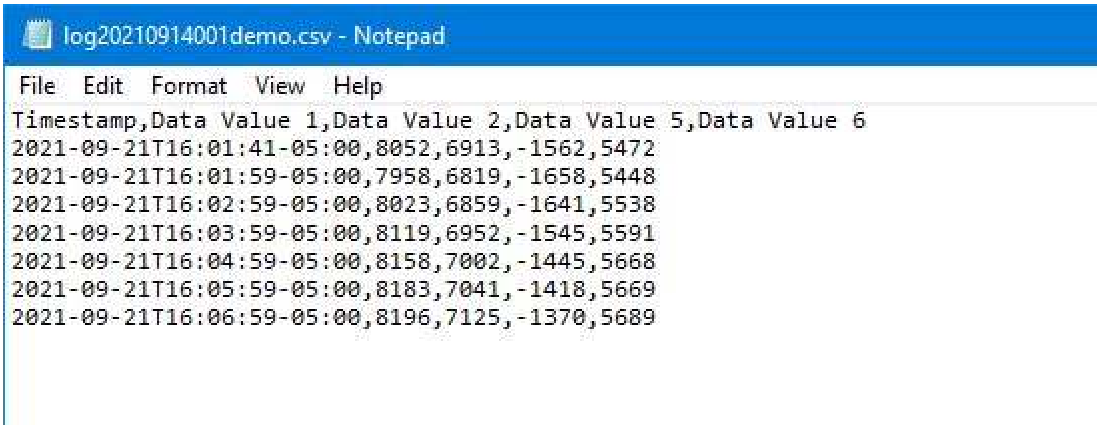
One of the many things you can do with data that has been published to the AWS server by the Babel Buster IoT Gateway is analyze and visualize the data. The graph illustrated below represents data published by an IoT Gateway, and the steps taken to get this graph are outlined later in this user guide.



The IoT Gateway is capable of more than just transferring data between Modbus devices and the AWS servers. 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 Babel Buster IoT Gateway 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.

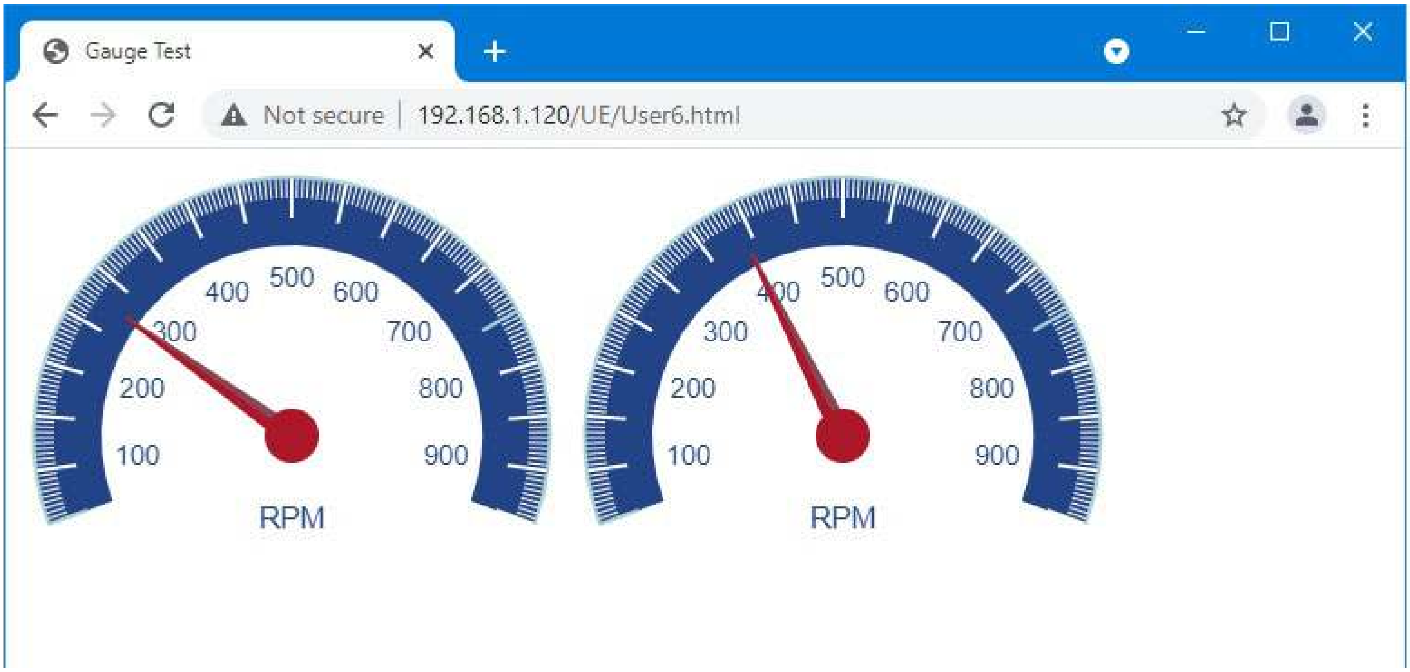


```
Timestamp,Data Value 1,Data Value 2,Data Value 5,Data Value 6
2021-09-21T16:01:41-05:00,8052,6913,-1562,5472
2021-09-21T16:01:59-05:00,7958,6819,-1658,5448
2021-09-21T16:02:59-05:00,8023,6859,-1641,5538
2021-09-21T16:03:59-05:00,8119,6952,-1545,5591
2021-09-21T16:04:59-05:00,8158,7002,-1445,5668
2021-09-21T16:05:59-05:00,8183,7041,-1418,5669
2021-09-21T16:06:59-05:00,8196,7125,-1370,5689
```

There are several features available for supporting control algorithms, and a real time scheduler is now included in the Babel Buster IoT Gateway. 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 IoT Gateway.

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 Babel Buster IoT Gateway 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 gateway as noted in the following section. Then set the IP address of your IoT Gateway, 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 IoT Gateway 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 IoT gateway 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 IoT Gateway to talk to AWS. Then move on to any of Sections 14 through 18 to cover the various ways you can interact with AWS. If you are not using AWS, you can skip Sections 13 through 18.

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.

• 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 19.

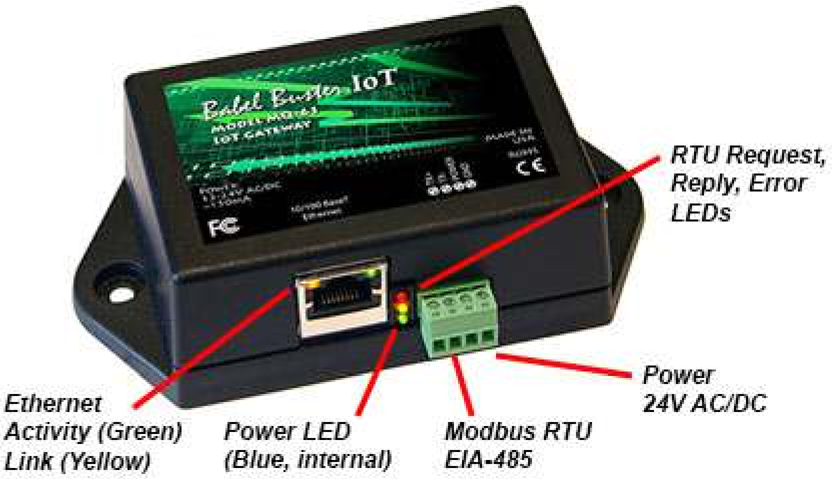
• Last but not least are a couple of advanced topics. If you want to create your own custom web pages to be served by the IoT Gateway's internal server, that is covered in Section 20. If you have an external server with an application that can use a REST API to query devices, the REST API is covered in section 21.

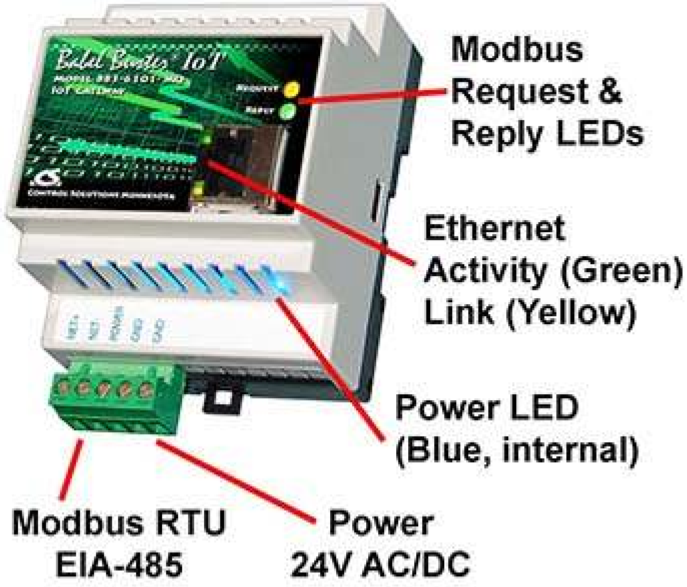
• Various details are covered in Appendix A through D. Be sure to look at the first 3 sections of Appendix A which cover hardware details you will need to be aware of. You can save the rest of the reference information for when you need it.

2.3 Connectors and Indicators

Follow these steps to make the initial connection to the Babel Buster MQ-61 or BB3-6101-MQ.

(a) Connect power. Apply +12 to +24VDC or 24VAC to the terminal marked “POWER”, and common or ground the terminal marked “GND”.


Babel Buster IoT
MODEL MQ-61
IOT GATEWAY
RTU Request, Reply, Error LEDs
Power 24V AC/DC
Ethernet Activity (Green) Link (Yellow)
Power LED (Blue, internal)
Modbus RTU EIA-485


Babel Buster IoT
Modbus Request & Reply LEDs
Ethernet Activity (Green) Link (Yellow)
Power LED (Blue, internal)
Modbus RTU EIA-485
Power 24V AC/DC

(b) Connect a CAT5 cable between the RJ-45 jack on the gateway, 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.

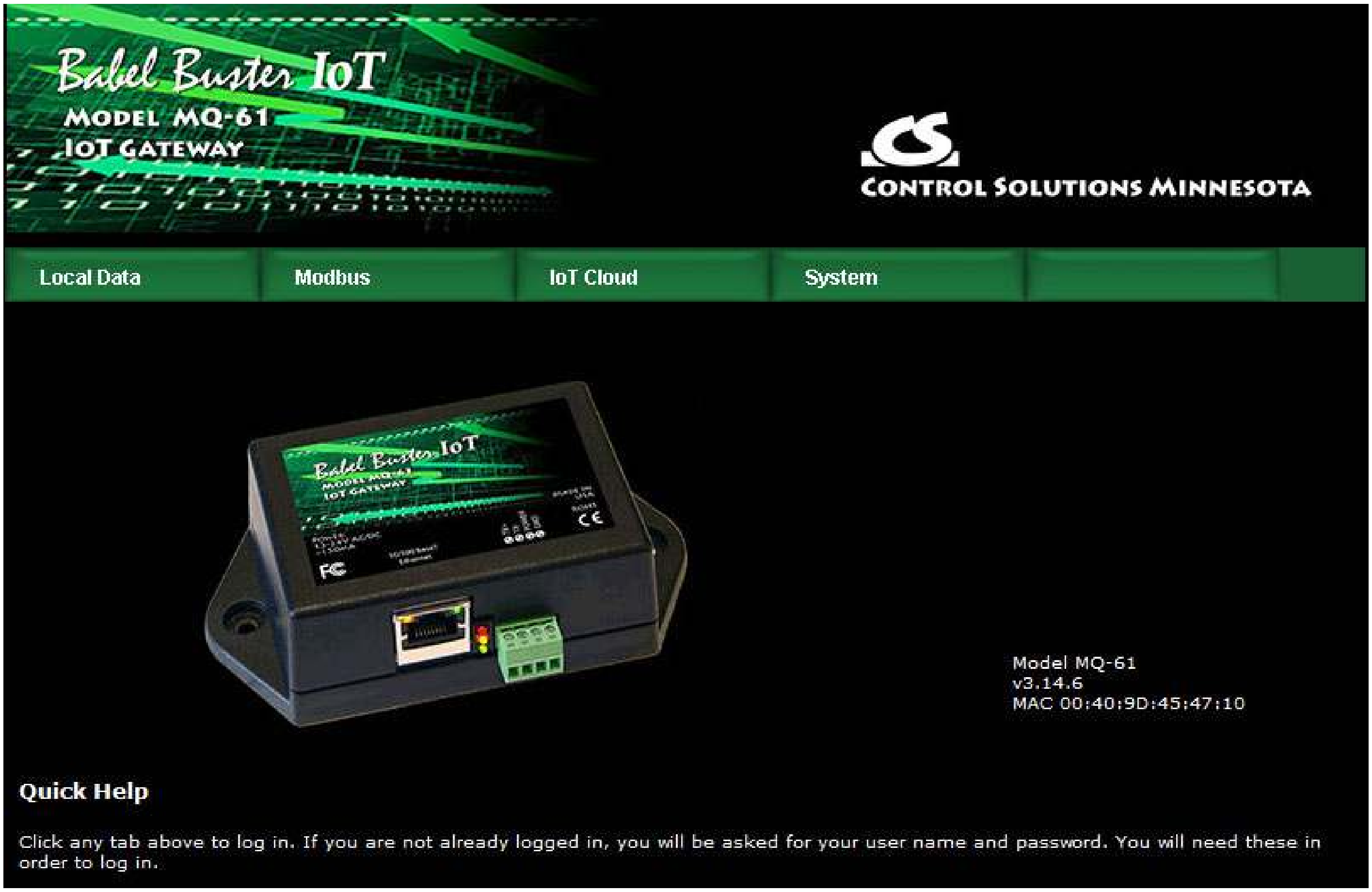
Ethernet link LED is the yellow LED integrated into the CAT5 connector. Ethernet activity LED is the green LED integrated into the CAT5 connector.

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

2.4 Open 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.



Model MQ-61
v3.14.6
MAC 00:40:9D:45:47:10

Quick Help

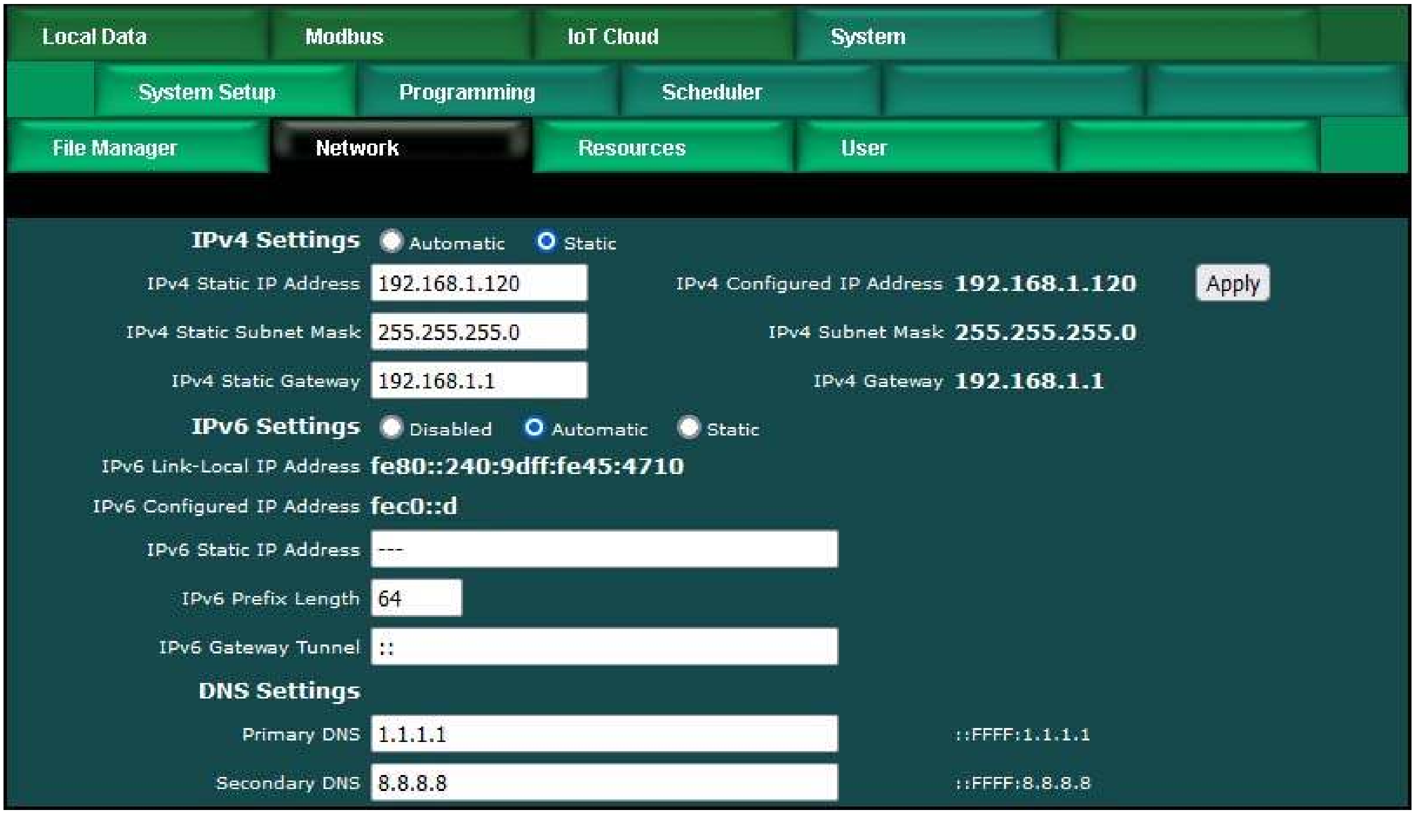
Click any tab above to log in. If you are not already logged in, you will be asked for your user name and password. You will need these in order to log in.

Open your browser, and enter “http://10.0.0.101/” in the address window. You should see a page with the “Babel Buster MQ-61” header shown above (or comparable page for BB3-6101-MQ). 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 IoT Gateway shipped, and unique to your IoT Gateway. 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.csimn.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 gateway, 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.



IPv4 Settings ○ Automatic ● Static

IPv4 Static IP Address: 192.168.1.120 — IPv4 Configured IP Address **192.168.1.120**

IPv4 Static Subnet Mask: 255.255.255.0 — IPv4 Subnet Mask **255.255.255.0**

IPv4 Static Gateway: 192.168.1.1 — 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: 64

IPv6 Gateway Tunnel: ::

DNS Settings

Primary DNS: 1.1.1.1 — ::FFFF:1.1.1.1

Secondary DNS: 8.8.8.8 — ::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.

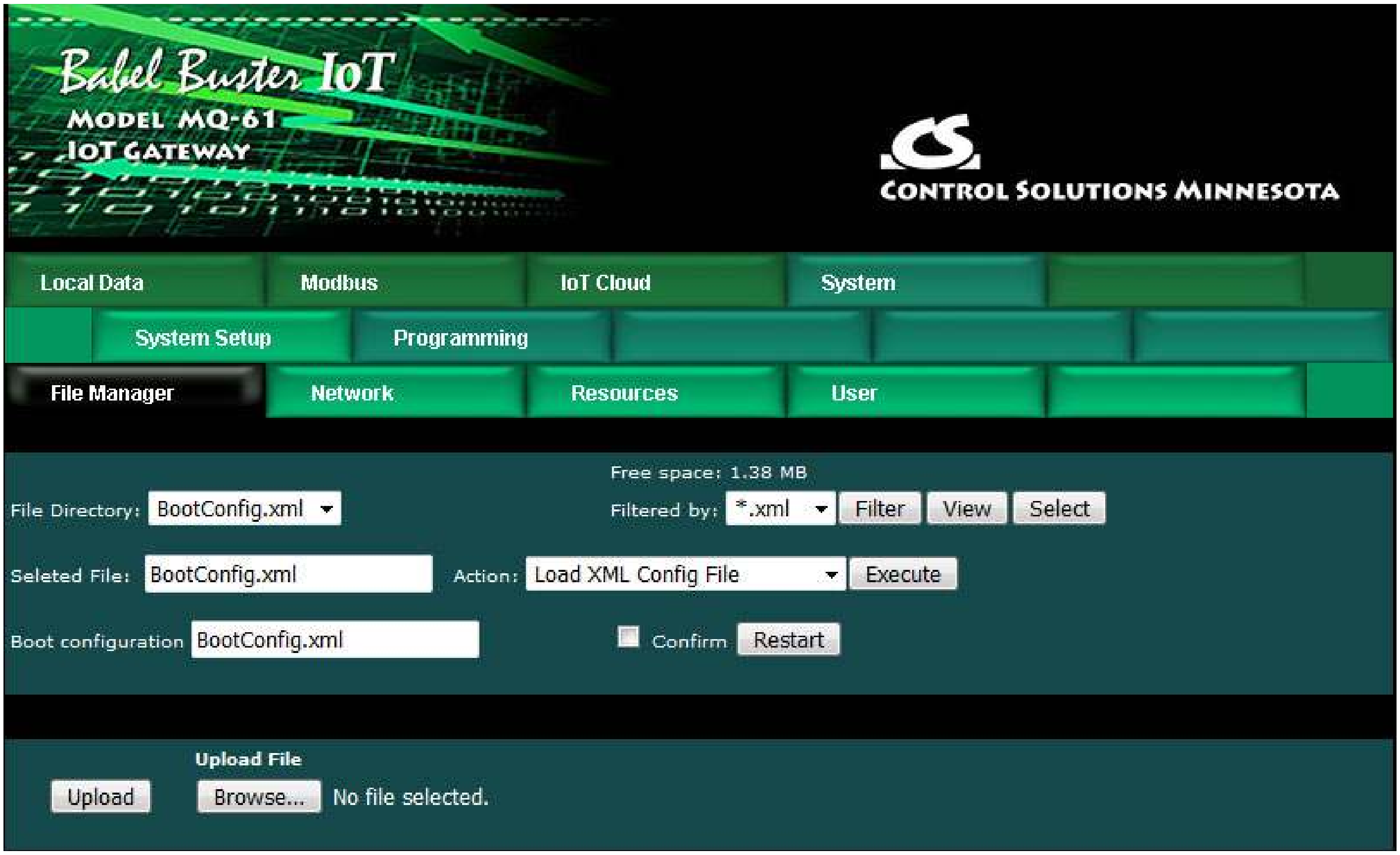
3. System Configuration and Resources

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 gateway 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 gateway. To retain those changes indefinitely (i.e. through restart or power cycle), you need to tell the gateway 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.



Free space: 1.38 MB

File Directory: BootConfig.xml

Filtered by: *.xml

Seleted File: BootConfig.xml

Action: Load XML Config File

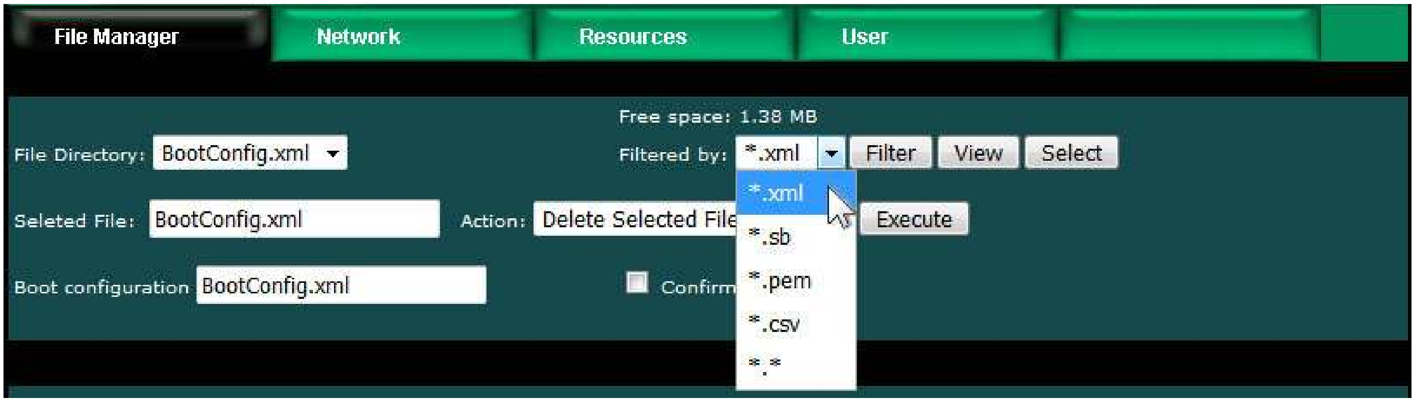
Boot configuration BootConfig.xml

□ Confirm

Upload File

No file selected.

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



Free space: 1.38 MB

File Directory: BootConfig.xml

Filtered by: *.xml

Seleted File: BootConfig.xml

Action: Delete Selected File

Boot configuration BootConfig.xml

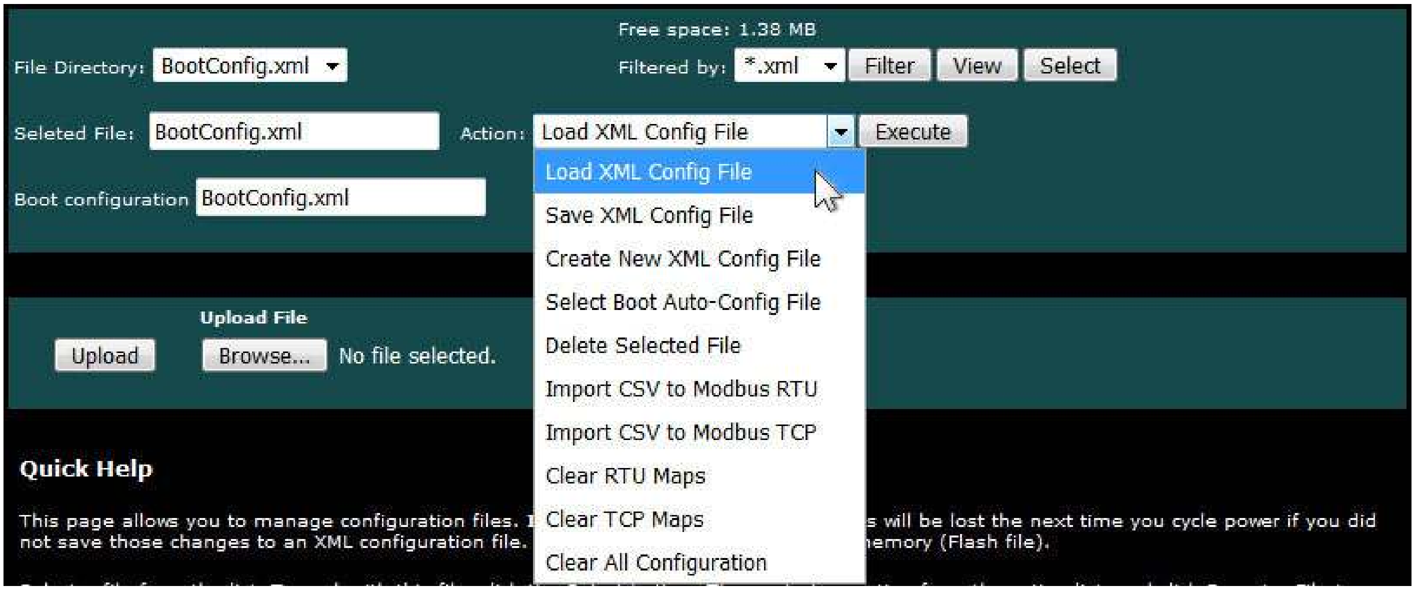
□ Confirm

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.



Free space: 1.38 MB

File Directory: BootConfig.xml

Filtered by: *.xml

Seleted File: BootConfig.xml

Action: Load XML Config File

Boot configuration BootConfig.xml

No file selected.

Quick Help

This page allows you to manage configuration files. I s will be lost the next time you cycle power if you did not save those changes to an XML configuration file. emory (Flash file).

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 gateway, 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 gateway, 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 Babel Buster 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 gateway'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 gateway 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 simple delete the file named as the Boot configuration file, and then restart or power cycle the IoT Gateway. 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 Babel Buster 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 gateway. Upon restart, and upon finding that the boot configuration name is BootConfig.xml, and it does not exist, the gateway will automatically create one with default parameters. The automatic creation of a default file will not occur with any other file name.

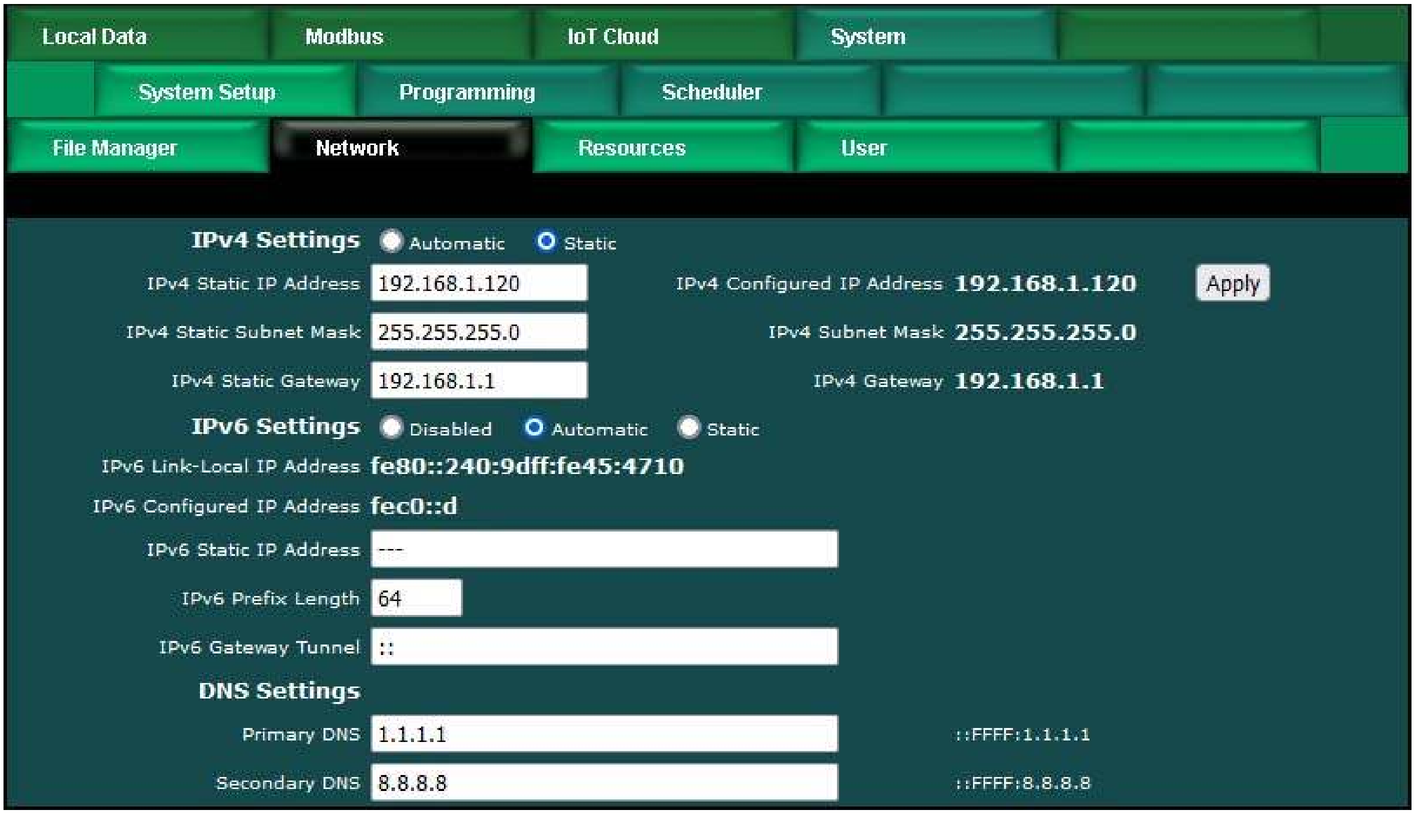
Manual Editing: It is possible to manually edit the XML file outside of the gateway. However, doing so is very prone to errors. If there are errors in the XML file, it will not load successfully on startup. If the configuration does not load on startup, none of the scanners will begin scanning. Because 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 gateway 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 Babel Busters 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 gateway, 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 Babel Buster's IP address as well as a few other important things.



IPv4 Settings ○ Automatic ● Static

IPv4 Static IP Address 192.168.1.120 | IPv4 Configured IP Address 192.168.1.120

IPv4 Static Subnet Mask 255.255.255.0 | IPv4 Subnet Mask 255.255.255.0

IPv4 Static Gateway 192.168.1.1 | 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 64

IPv6 Gateway Tunnel ::

DNS Settings

Primary DNS 1.1.1.1 ::FFFF:1.1.1.1

Secondary DNS 8.8.8.8 ::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.

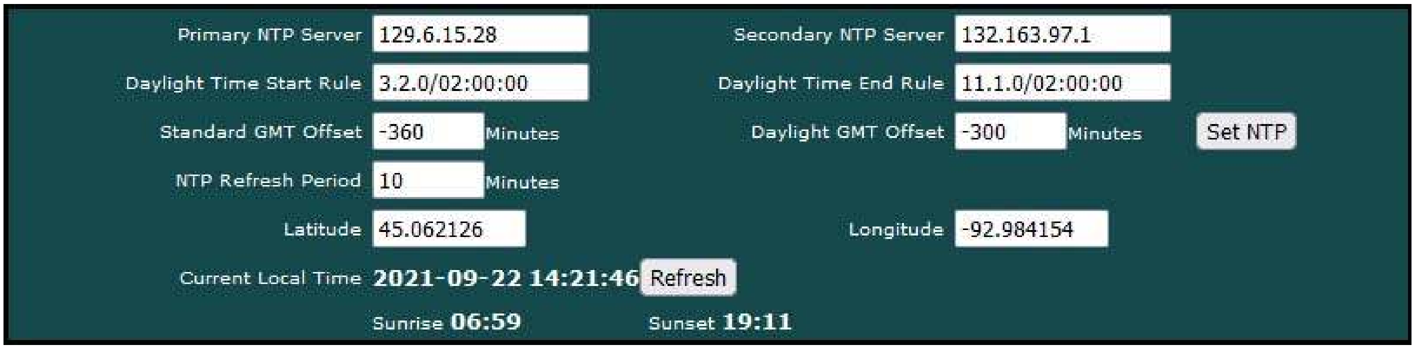
The IPv6 static IP address window is the configured static address. If "Static" is selected and a new IP address entered as the static address, this new address will not take effect until the next system restart.

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 gateway 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 Babel Buster maintains time and date via SNTP services.



Primary NTP Server 129.6.15.28
Secondary NTP Server 132.163.97.1
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 10 Minutes
Latitude 45.062126
Longitude -92.984154
Current Local Time 2021-09-22 14:21:46
Sunrise 06:59
Sunset 19:11

NTP setup: Enter a primary and secondary IP address of NTP servers, such as those found at wwv.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 Babel Buster 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 <= m <= 12)

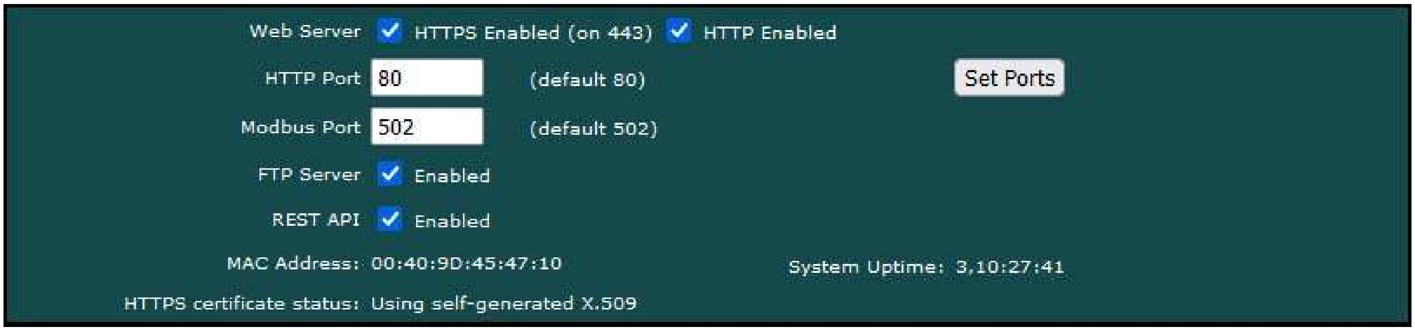
n indicates which week of the month (1 <= n <= 5). 5 = the last week in the month.

d indicates what day of the week (0 <= d <= 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.

3.3.3 Port Settings



Web Server ☑ HTTPS Enabled (on 443) ☑ HTTP Enabled

HTTP Port 80 (default 80)

Modbus Port 502 (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 gateway to make that new port take effect. Don't forget to append the port number to the gateway'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 gateway 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 Babel Buster before they will take effect.

3.4 Resource Allocation

Note: The Resource Allocation procedure has changed starting in firmware version 3.16.1 as compared to 3.15.X.

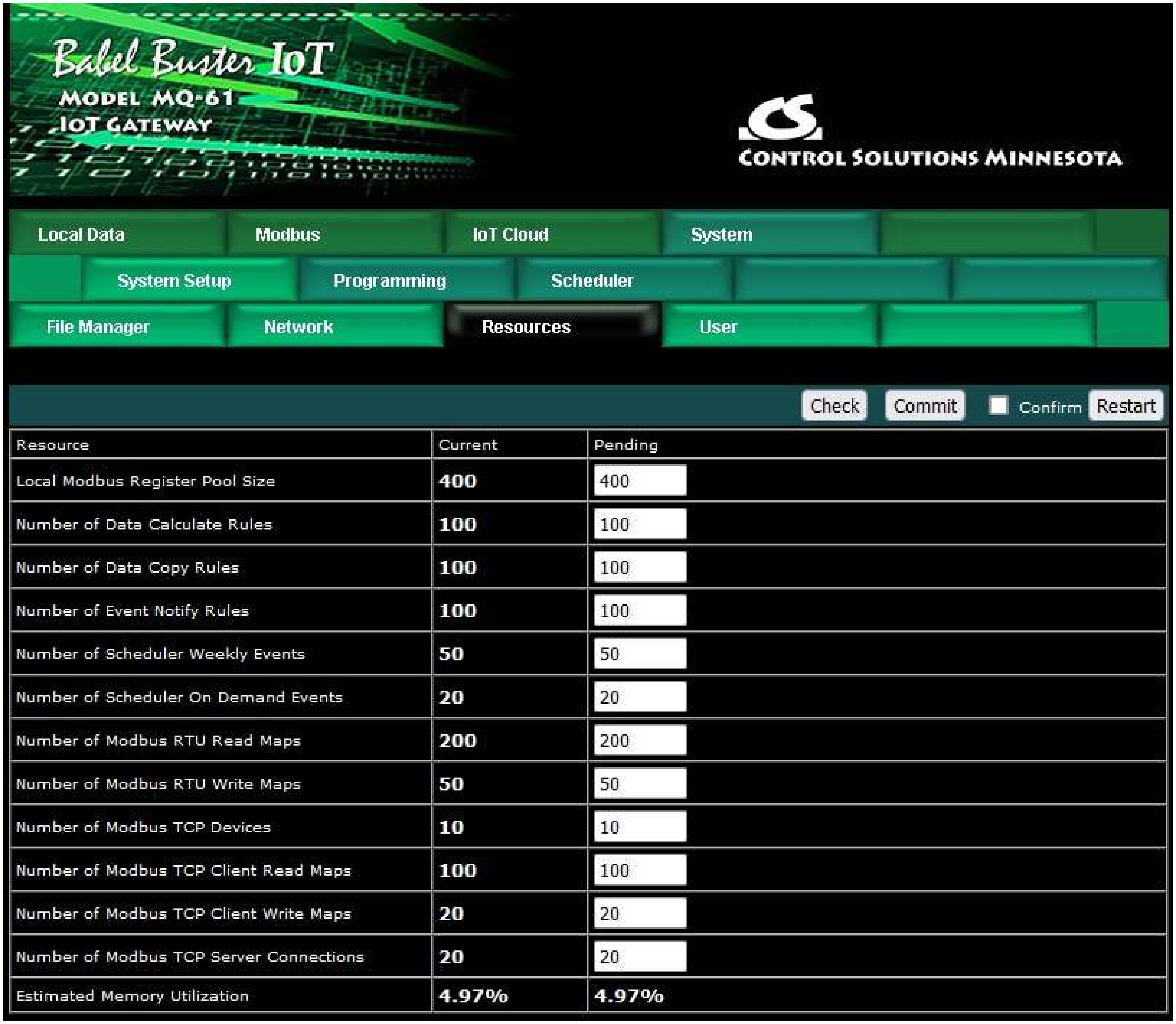
Historically, Control Solutions gateways had a fixed set of resources to work with. Invariably, there were always users that wanted less of this and more of that. Therefore, while there are still maximums imposed, you can now shift resources around as best suits your application. An example is shown 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	400
Number of Data Calculate Rules	100	100
Number of Data Copy Rules	100	100
Number of Event Notify Rules	100	100
Number of Scheduler Weekly Events	50	50
Number of Scheduler On Demand Events	20	20
Number of Modbus RTU Read Maps	200	200
Number of Modbus RTU Write Maps	50	50
Number of Modbus TCP Devices	10	10
Number of Modbus TCP Client Read Maps	100	100
Number of Modbus TCP Client Write Maps	20	20
Number of Modbus TCP Server Connections	20	20
Estimated Memory Utilization	4.97%	4.97%

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.

3.5 User Login Passwords

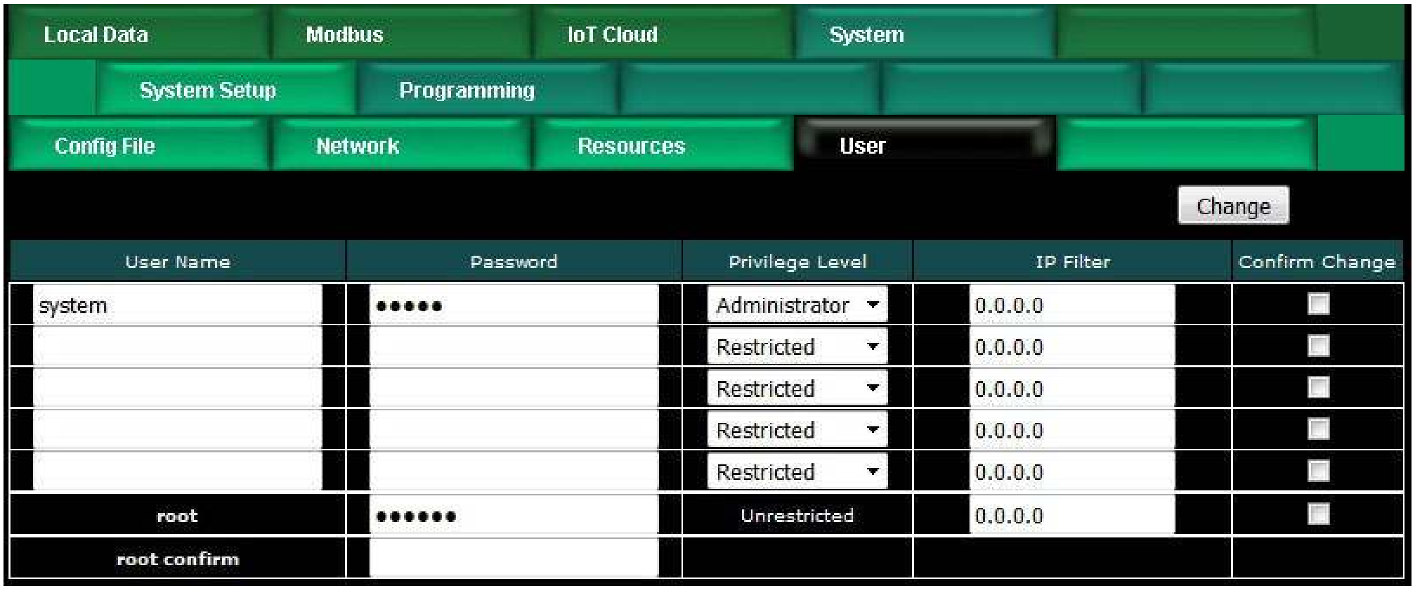
There is only one default login provided initially. That login is the username "root" and root's password is a unique password generated specifically for this particular Babel Buster gateway. That unique password was provided for you in documentation included with the shipment. That unique password complies 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 has no meaning in the Babel Buster (other than block access to everything) since it does not operate as a user defined web server.

You also have the option of IP filtering. If set, then the user can only access Babel Buster'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.



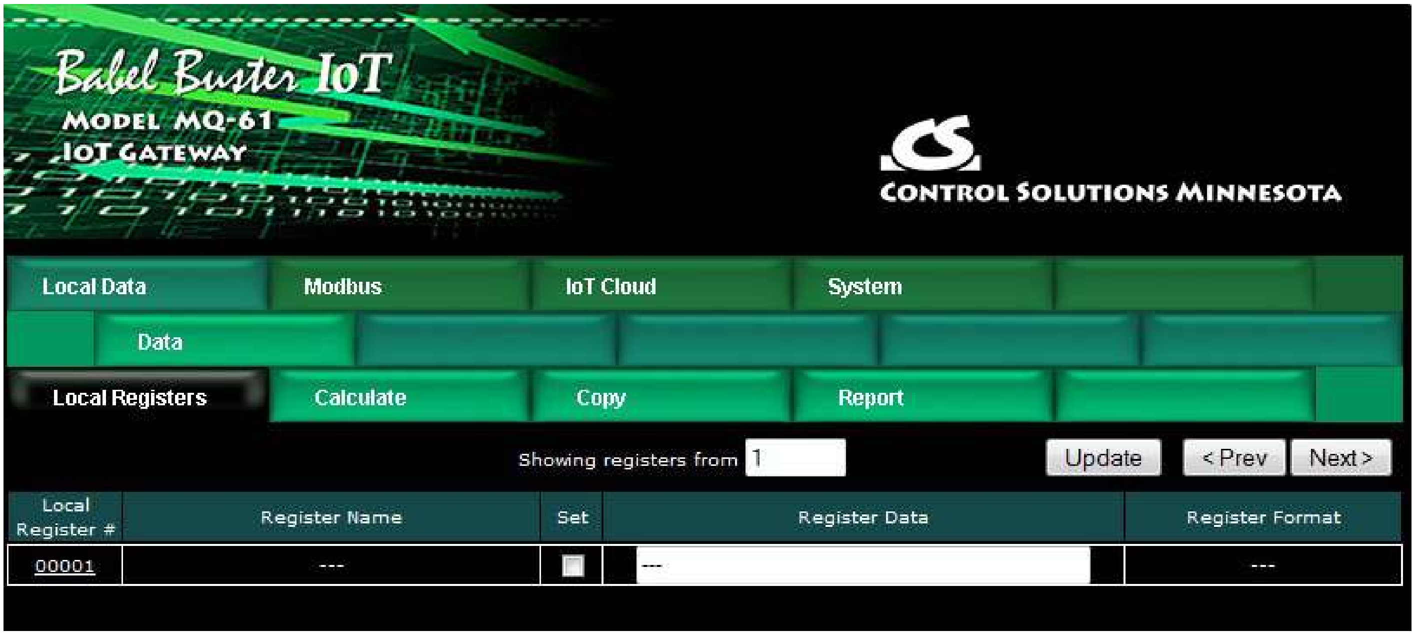
User Name	Password	Privilege Level	IP Filter	Confirm Change
system	•••••	Administrator	0.0.0.0	□
		Restricted	0.0.0.0	□
		Restricted	0.0.0.0	□
		Restricted	0.0.0.0	□
		Restricted	0.0.0.0	□
root	••••••	Unrestricted	0.0.0.0	□
root confirm				

4. Configuring Local Registers

4.1 Creating Local Registers

Babel Buster gateways 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 Babel Buster IoT Gateway out of the box, it has no registers. You get to create registers as you need them, and several data formats are supported, from 16-bit to 64-bit, both integer and floating point.

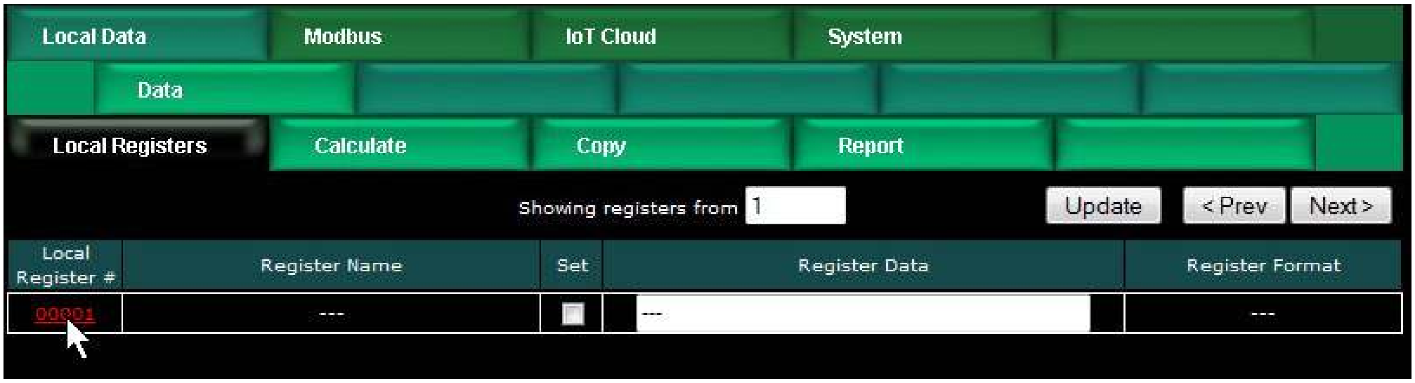
Your first visit to the Local Registers page will be the uneventful display illustrated below.



Showing registers from 1

Local Register #	Register Name	Set	Register Data	Register Format
00001	---	□	---	---

To begin the process of creating registers, click on the only register number available at this point. 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.



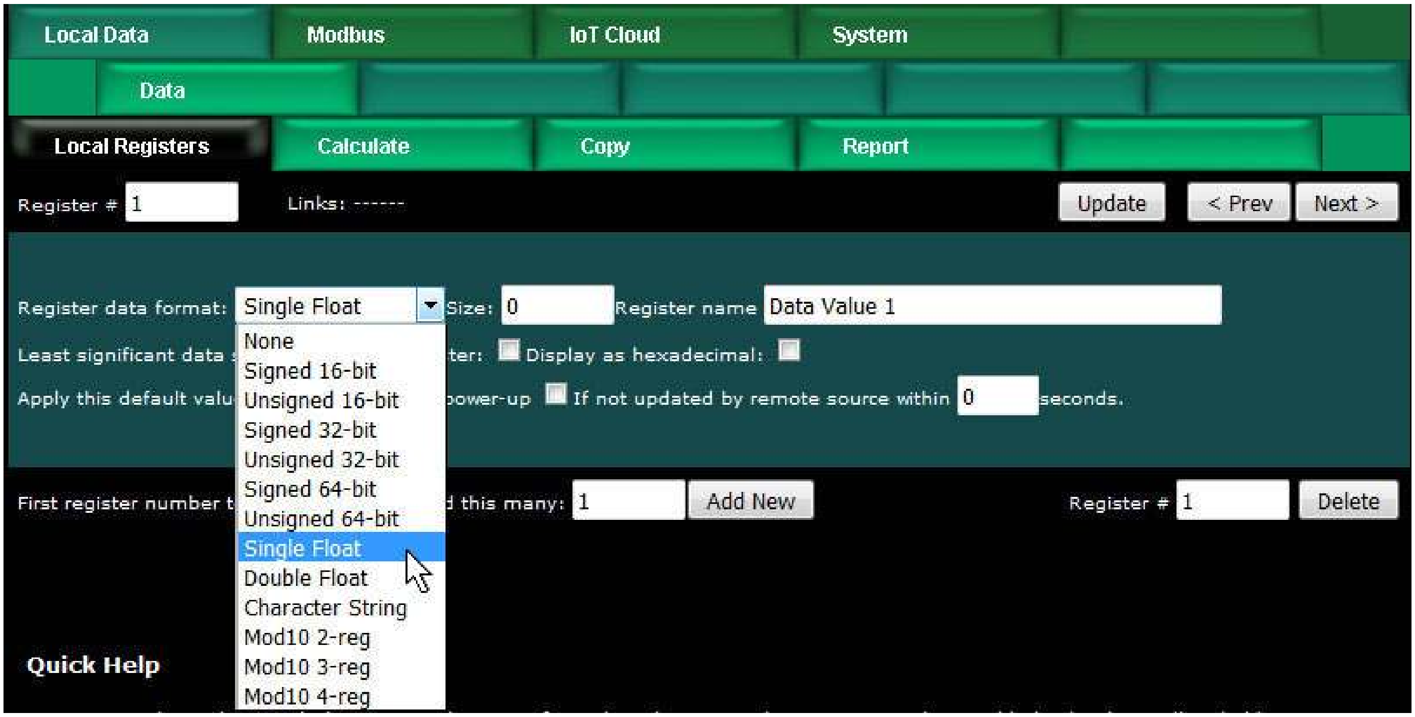
Showing registers from 1

Local Register #	Register Name	Set	Register Data	Register Format
00001	---	□	---	---

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 Babel Buster as anything bigger than 16 bits, it is actually a pair (or series of 2 or more) 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.



Register # 1 Links: ------

Register data format: Single Float Size: 0 Register name Data Value 1

Least significant data s ter: □ Display as hexadecimal: □

Apply this default valu power-up □ If not updated by remote source within 0 seconds.

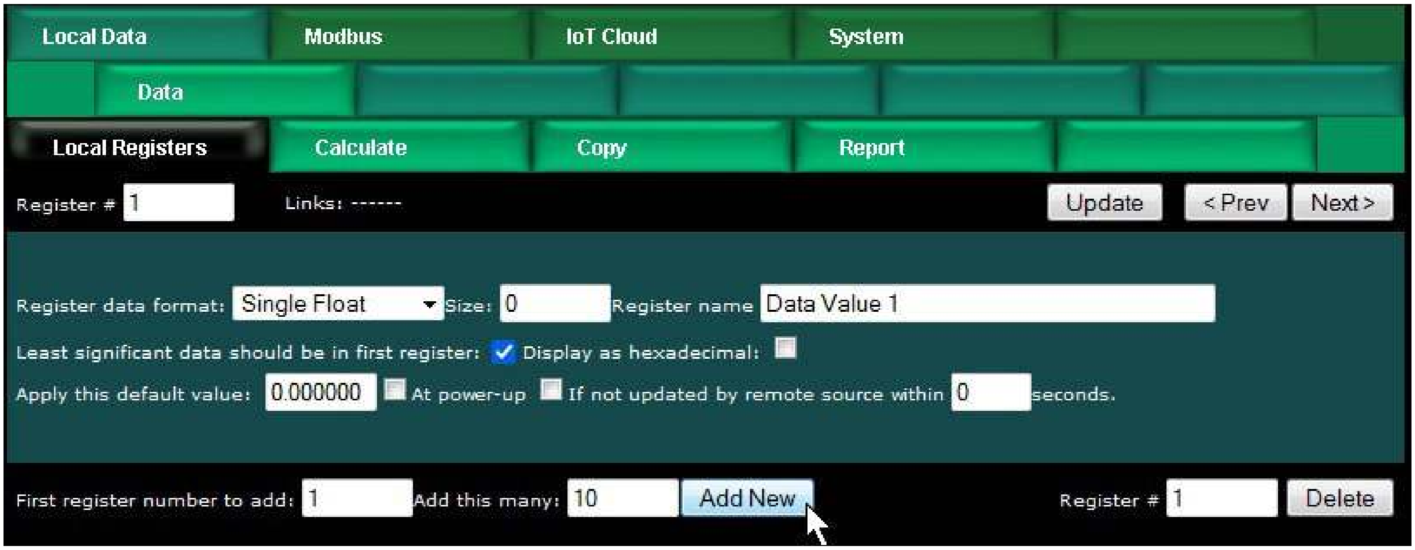
First register number t d this many: 1 Register # 1

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. Babel Buster 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.



Register # 1 Links: ------

Register data format: Single Float Size: 0 Register name Data Value 1

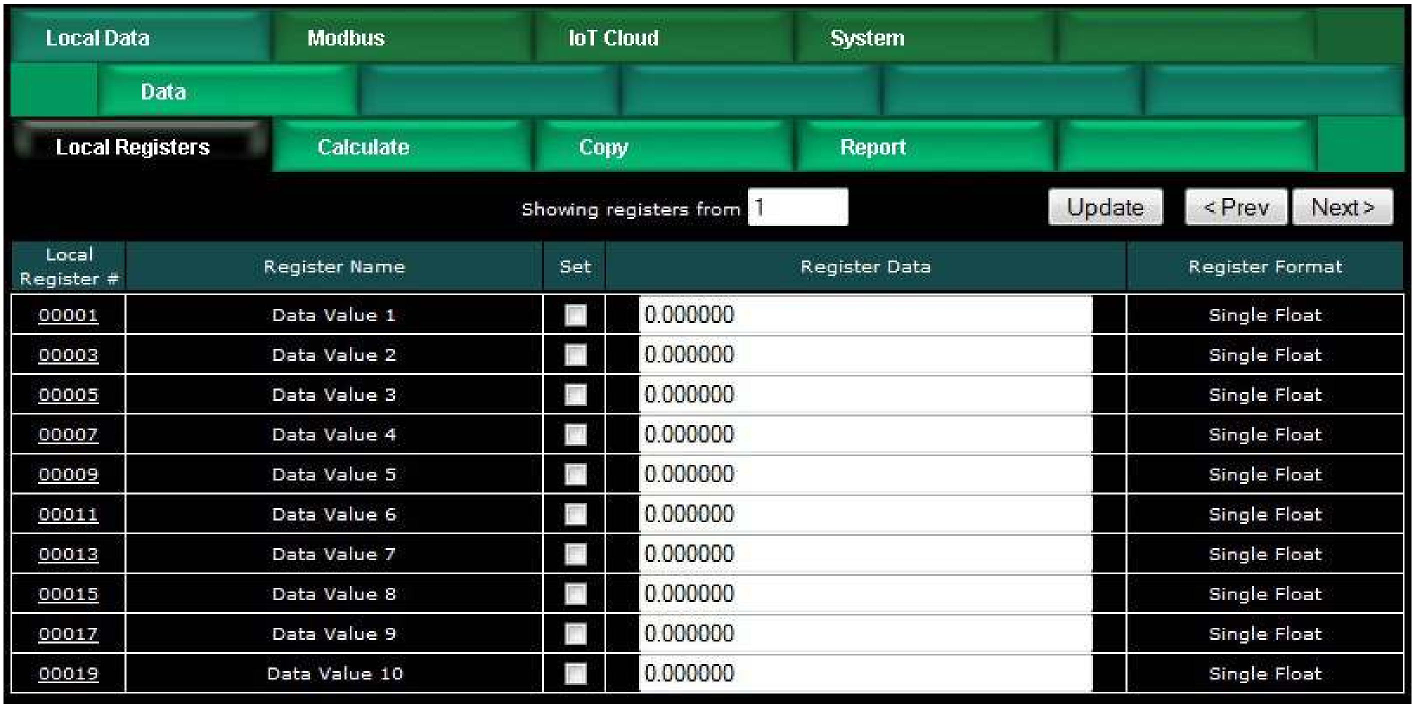
Least significant data should be in first register: ☑ Display as hexadecimal: □

Apply this default value: 0.000000 □ At power-up □ If not updated by remote source within 0 seconds.

First register number to add: 1 Add this many: 10 Register # 1

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.

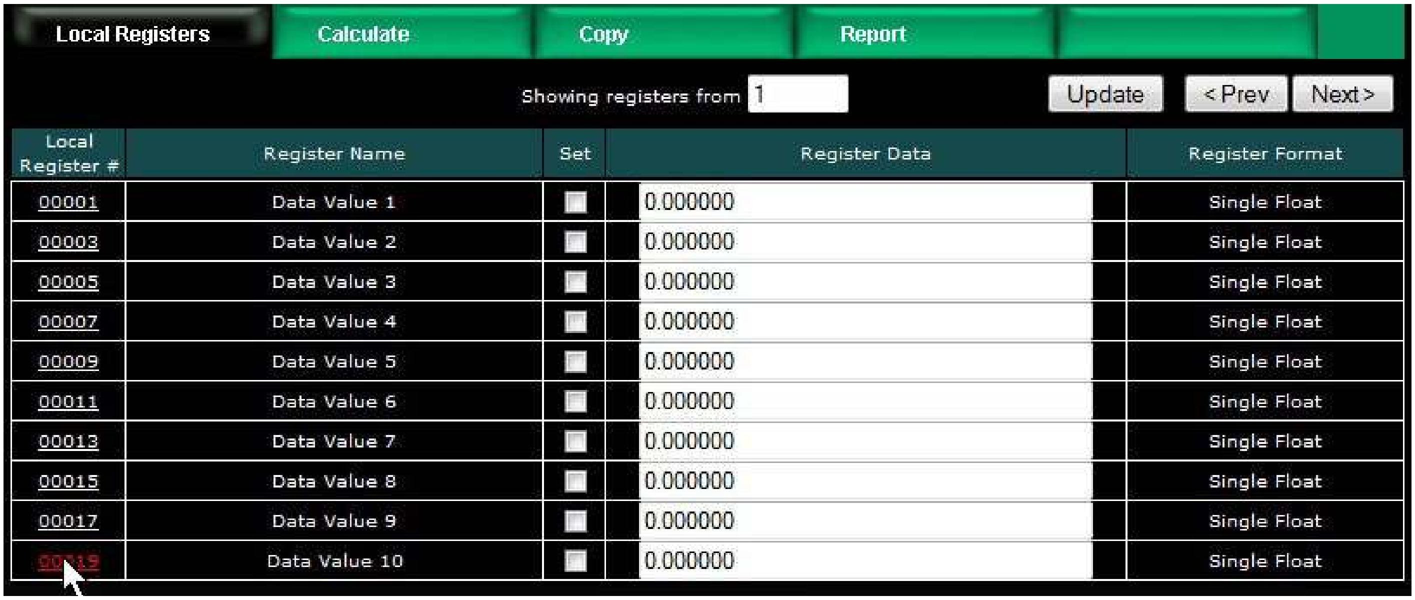
Note that the name given in the screen above was "Data Value 1" but our series of 10 new registers came up with 10 sequential names. If the last thing to appear in the name given when assigning new registers is a number, this value will be incremented by one in the name of each successive new register.



Showing registers from 1

Local Register #	Register Name	Set	Register Data	Register Format
00001	Data Value 1	□	0.000000	Single Float
00003	Data Value 2	□	0.000000	Single Float
00005	Data Value 3	□	0.000000	Single Float
00007	Data Value 4	□	0.000000	Single Float
00009	Data Value 5	□	0.000000	Single Float
00011	Data Value 6	□	0.000000	Single Float
00013	Data Value 7	□	0.000000	Single Float
00015	Data Value 8	□	0.000000	Single Float
00017	Data Value 9	□	0.000000	Single Float
00019	Data Value 10	□	0.000000	Single Float

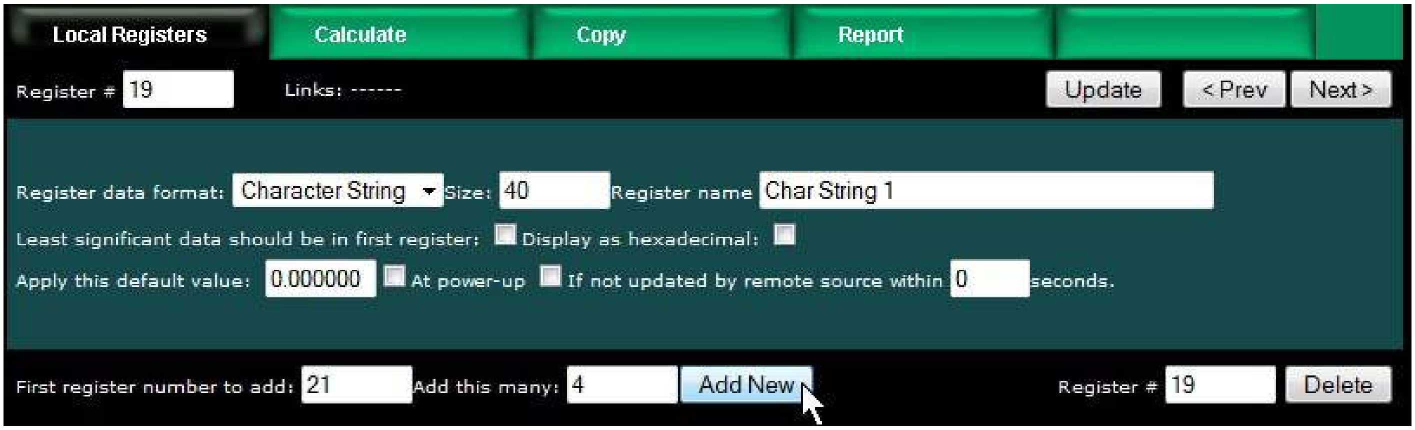
Now let's proceed to add some character strings. Click on the last register number assigned thus far to open the register detail page.



Showing registers from 1

Local Register #	Register Name	Set	Register Data	Register Format
00001	Data Value 1	☐	0.000000	Single Float
00003	Data Value 2	☐	0.000000	Single Float
00005	Data Value 3	☐	0.000000	Single Float
00007	Data Value 4	☐	0.000000	Single Float
00009	Data Value 5	☐	0.000000	Single Float
00011	Data Value 6	☐	0.000000	Single Float
00013	Data Value 7	☐	0.000000	Single Float
00015	Data Value 8	☐	0.000000	Single Float
00017	Data Value 9	☐	0.000000	Single Float
00019	Data Value 10	☐	0.000000	Single Float

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 Babel Buster.



Register # 19 Links: ------

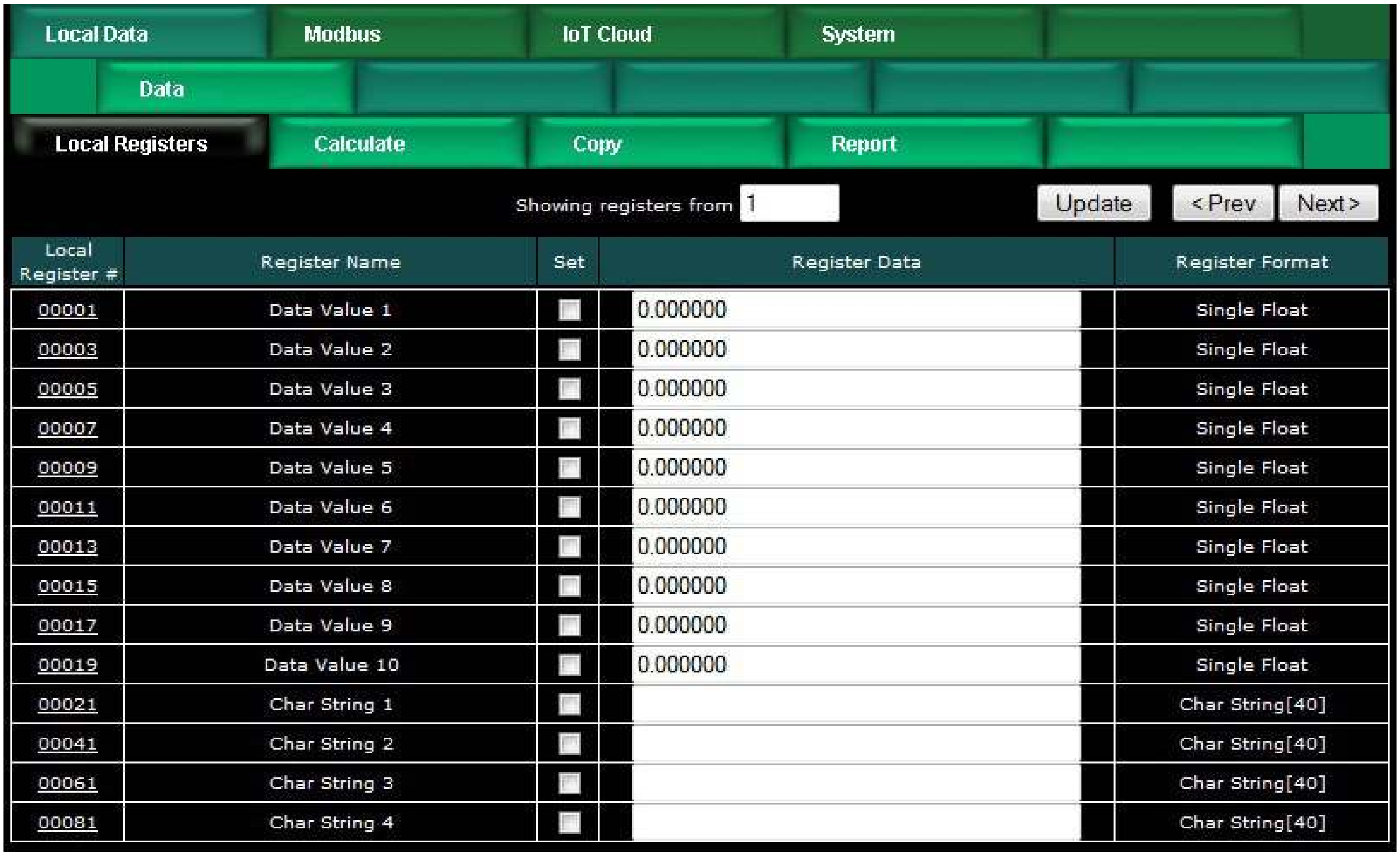
Register data format: Character String Size: 40 Register name Char String 1

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

Apply this default value: 0.000000 ☐ At power-up ☐ If not updated by remote source within 0 seconds.

First register number to add: 21 Add this many: 4 Register # 19

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.

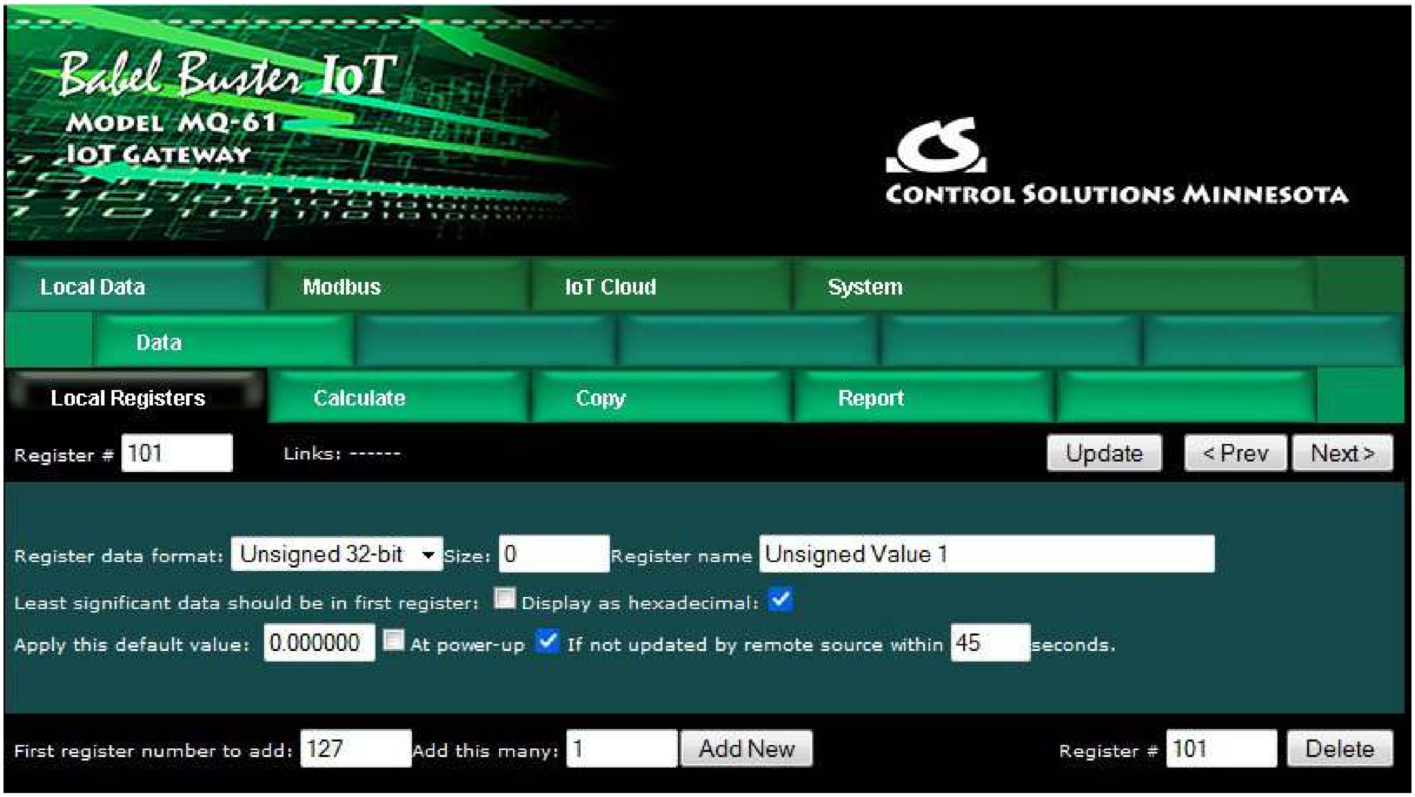


Showing registers from 1

Local Register #	Register Name	Set	Register Data	Register Format
00001	Data Value 1	☐	0.000000	Single Float
00003	Data Value 2	☐	0.000000	Single Float
00005	Data Value 3	☐	0.000000	Single Float
00007	Data Value 4	☐	0.000000	Single Float
00009	Data Value 5	☐	0.000000	Single Float
00011	Data Value 6	☐	0.000000	Single Float
00013	Data Value 7	☐	0.000000	Single Float
00015	Data Value 8	☐	0.000000	Single Float
00017	Data Value 9	☐	0.000000	Single Float
00019	Data Value 10	☐	0.000000	Single Float
00021	Char String 1	☐		Char String[40]
00041	Char String 2	☐		Char String[40]
00061	Char String 3	☐		Char String[40]
00081	Char String 4	☐		Char String[40]

4.2 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.



Register # 101 Links: ------

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

Least significant data should be in first register: ☐ Display as hexadecimal: ☑

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 Register # 101

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 Babel Buster gateway 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.

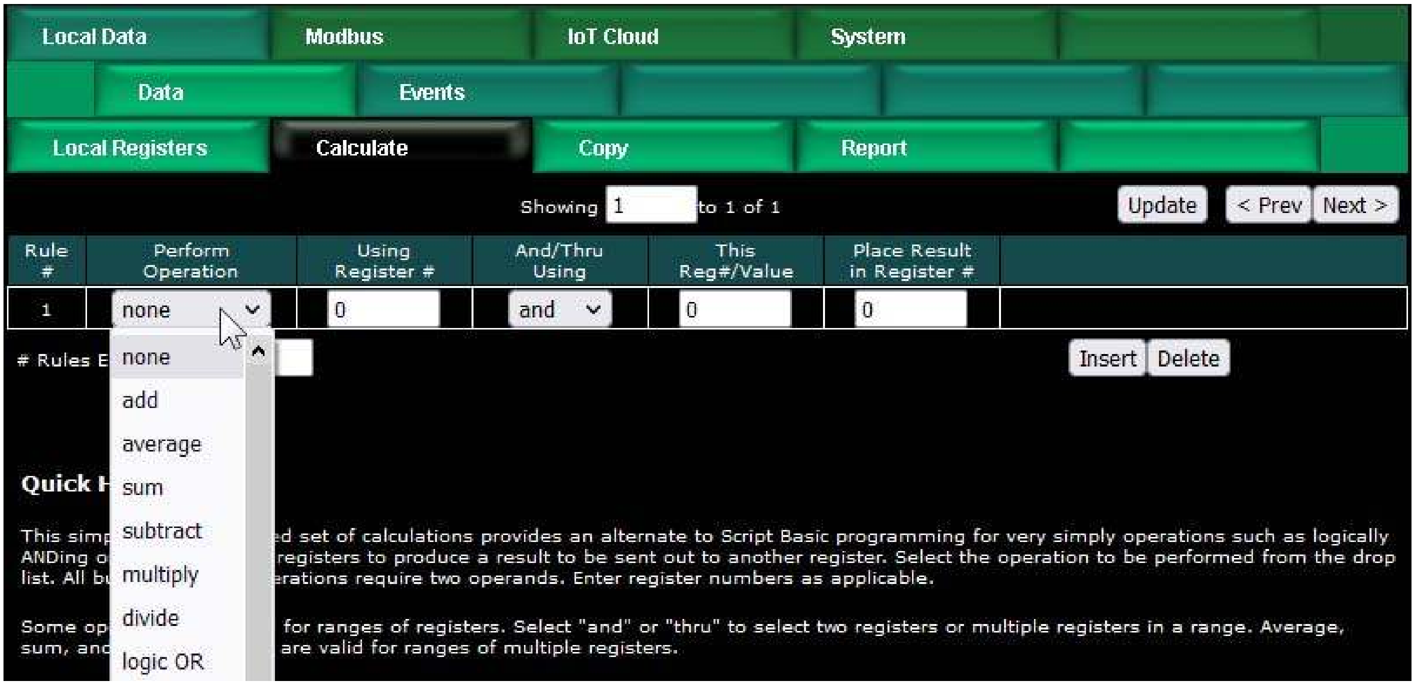
Applying a default at power-up is potentially useful if you want this gateway to always write a certain value to some other Modbus slave any time the system wakes up.

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.

4.3 Local Register Calculate Rules

The Babel Buster IoT Gateway 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 IoT 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.



Showing 1 to 1 of 1

Rule #	Perform Operation	Using Register #	And/Thru Using	This Reg#/Value	Place Result in Register #
1	none	0	and	0	0

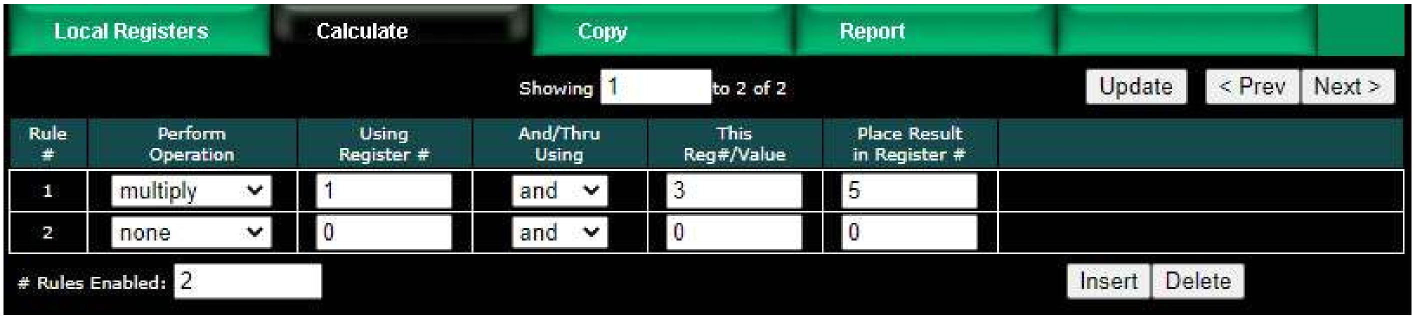
Rules E

Quick H

This simp d set of calculations provides an alternate to Script Basic programming for very simply operations such as logically
ANDing o registers to produce a result to be sent out to another register. Select the operation to be performed from the drop
list. All b erations require two operands. Enter register numbers as applicable.

Some op for ranges of registers. Select "and" or "thru" to select two registers or multiple registers in a range. Average,
sum, and are valid for ranges of multiple registers.

Here is an example of a template rule that multiplies register 1 by register 3 and places the result in register 5.

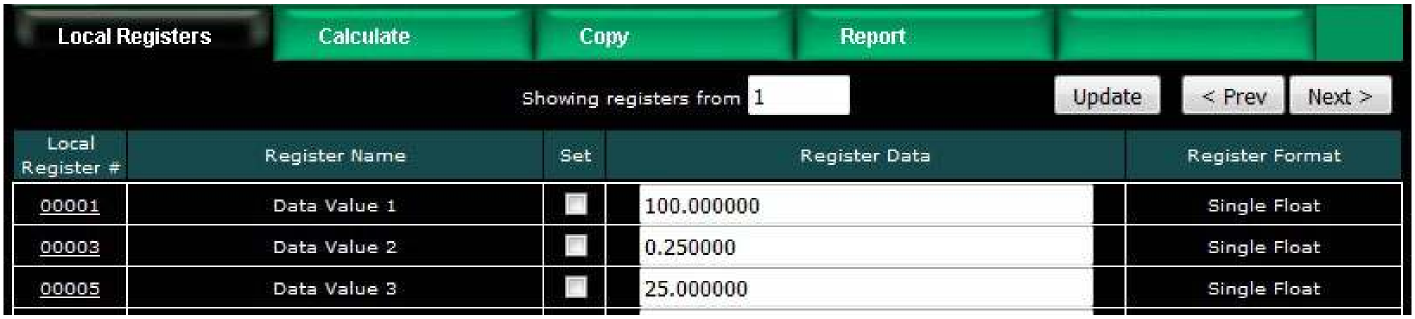


Showing 1 to 2 of 2

Rule #	Perform Operation	Using Register #	And/Thru Using	This Reg#/Value	Place Result in Register #
1	multiply	1	and	3	5
2	none	0	and	0	0

Rules Enabled: 2

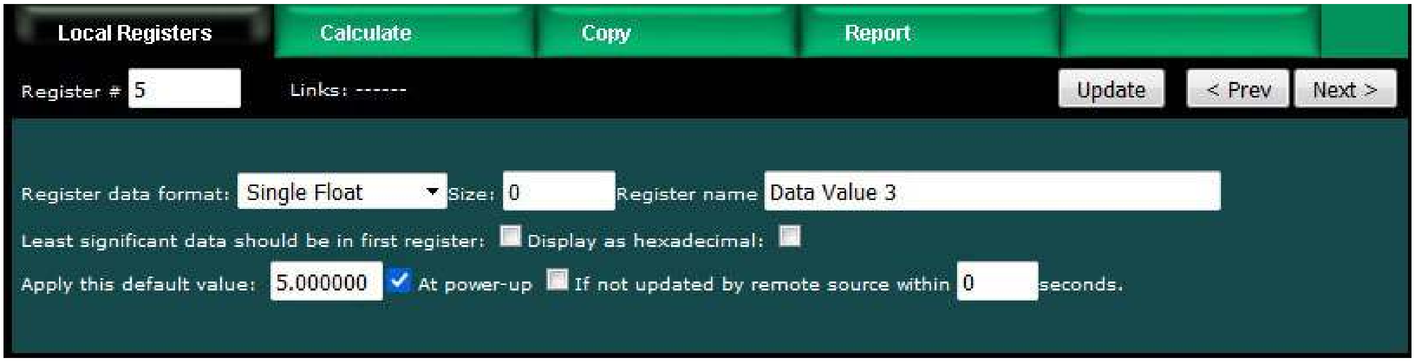
If registers 1 and 3 contain the values shown below, then the result shown in register 5 would be expected.



Showing registers from 1

Local Register #	Register Name	Set	Register Data	Register Format
00001	Data Value 1	□	100.000000	Single Float
00003	Data Value 2	□	0.250000	Single Float
00005	Data Value 3	□	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.



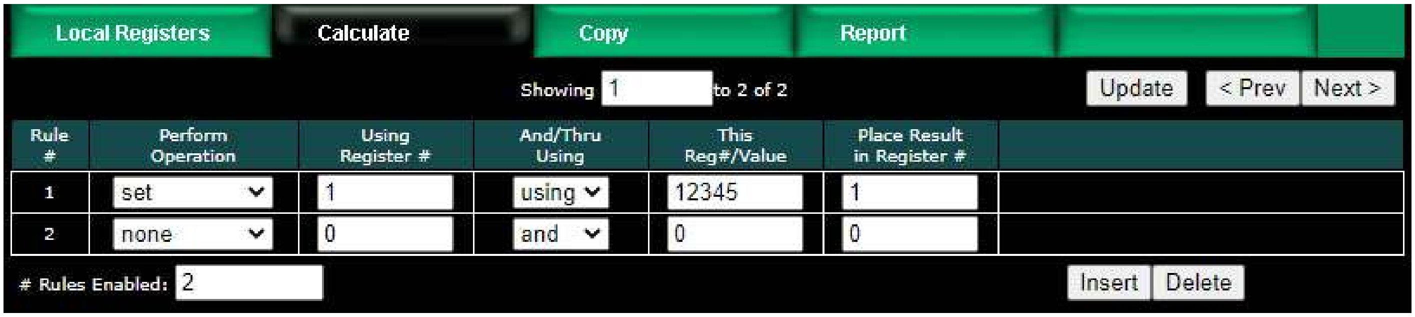
Register # 5 Links: ------

Register data format: Single Float Size: 0 Register name Data Value 3

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

Apply this default value: 5.000000 ☑ At power-up ☐ If not updated by remote source within 0 seconds.

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 1 to a value of 12345.



Showing 1 to 2 of 2

Rule #	Perform Operation	Using Register #	And/Thru Using	This Reg#/Value	Place Result in Register #
1	set	1	using	12345	1
2	none	0	and	0	0

Rules Enabled: 2

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

add	Add two registers
average	Average two or more registers
sum	Sum two or more registers
subtract	Subtract second register from first
multiply	Multiply two registers
divide	Divide first register by second
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)

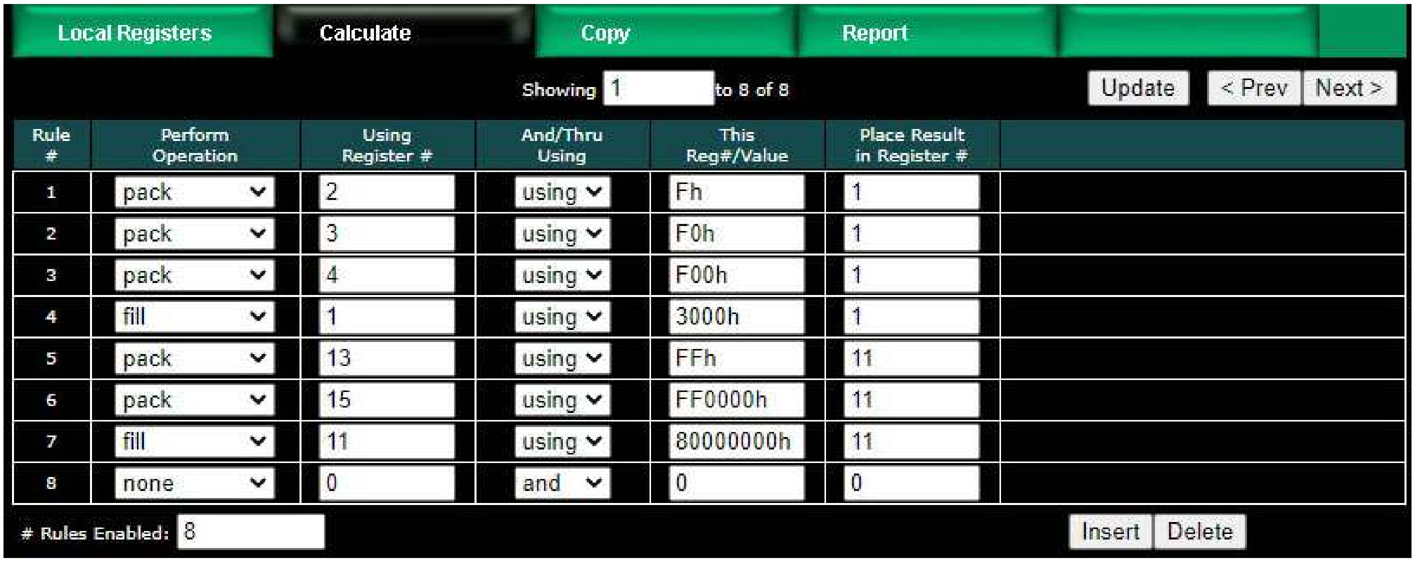
Operations "using" a given value will have an unsigned integer value in the "This Reg#/Value" column rather than a register number. These values will be displayed as integer for most operations, but will be displayed in hexadecimal for pack, fill, and unpack operations since these operate primarily on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

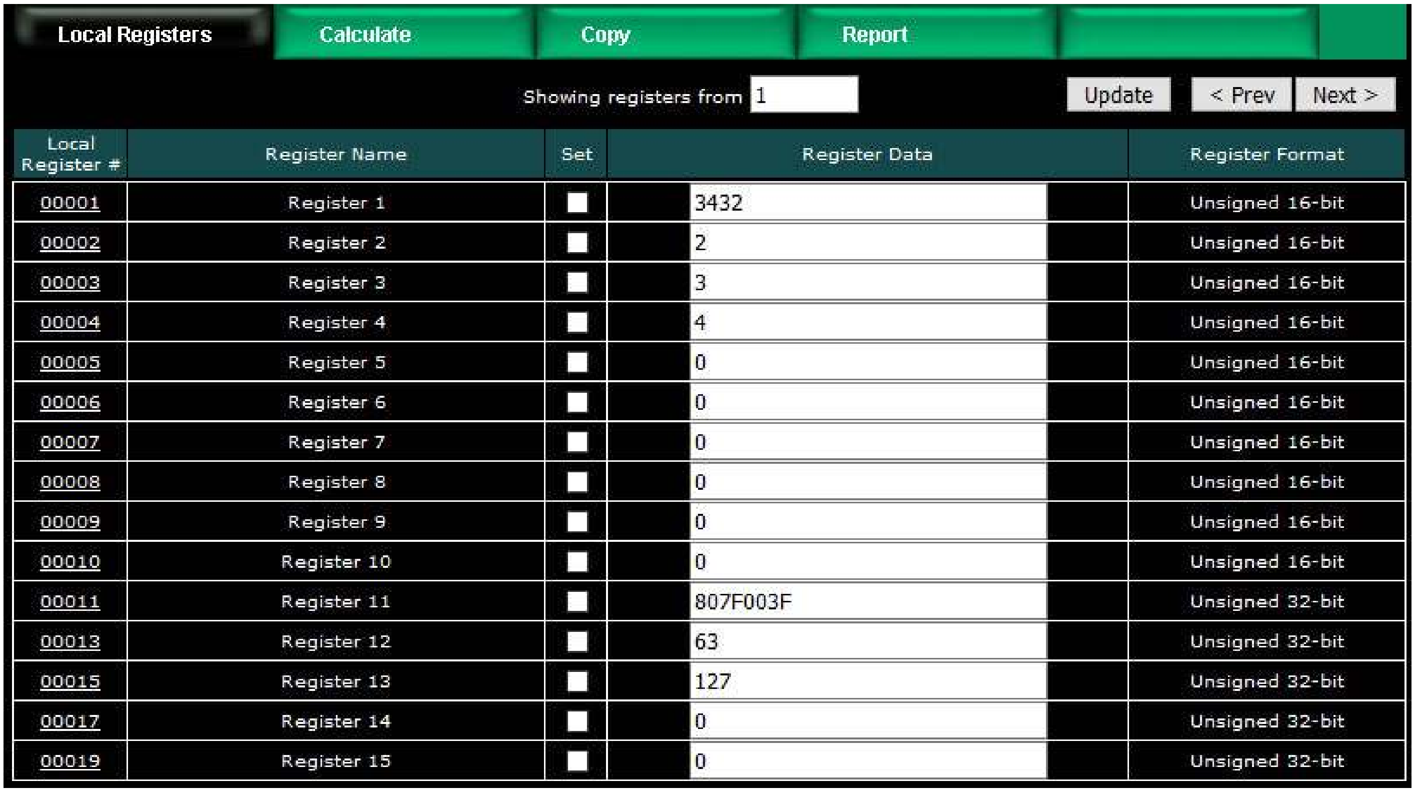
Fill is simple - it simply logically OR's the bit mask into the result register.



Rule #	Perform Operation	Using Register #	And/Thru Using	This Reg#/Value	Place Result in Register #
1	pack	2	using	Fh	1
2	pack	3	using	F0h	1
3	pack	4	using	F00h	1
4	fill	1	using	3000h	1
5	pack	13	using	FFh	11
6	pack	15	using	FF0000h	11
7	fill	11	using	80000000h	11
8	none	0	and	0	0

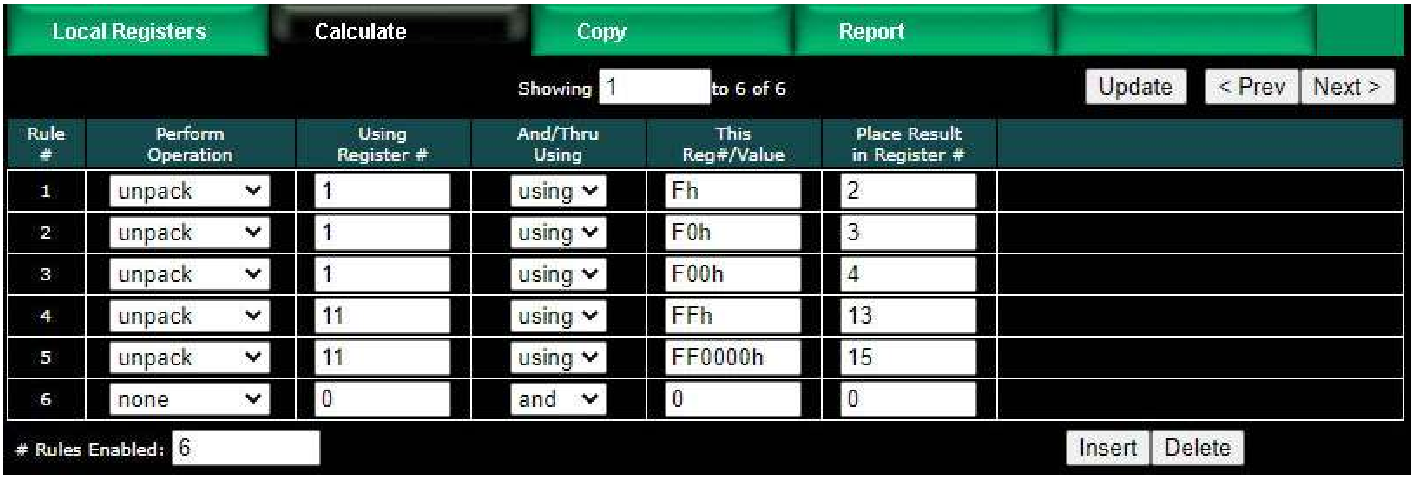
Rules Enabled: 8

The example below shows the content of result registers 1 and 11 using the above calculate rules and the local register values shown. The contents of registers 2, 3, and 4 are packed into register 1. The contents of registers 13 and 15 are packed into 11 (all 32-bit register pairs).



Local Register #	Register Name	Set	Register Data	Register Format
00001	Register 1	□	3432	Unsigned 16-bit
00002	Register 2	□	2	Unsigned 16-bit
00003	Register 3	□	3	Unsigned 16-bit
00004	Register 4	□	4	Unsigned 16-bit
00005	Register 5	□	0	Unsigned 16-bit
00006	Register 6	□	0	Unsigned 16-bit
00007	Register 7	□	0	Unsigned 16-bit
00008	Register 8	□	0	Unsigned 16-bit
00009	Register 9	□	0	Unsigned 16-bit
00010	Register 10	□	0	Unsigned 16-bit
00011	Register 11	□	807F003F	Unsigned 32-bit
00013	Register 12	□	63	Unsigned 32-bit
00015	Register 13	□	127	Unsigned 32-bit
00017	Register 14	□	0	Unsigned 32-bit
00019	Register 15	□	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).

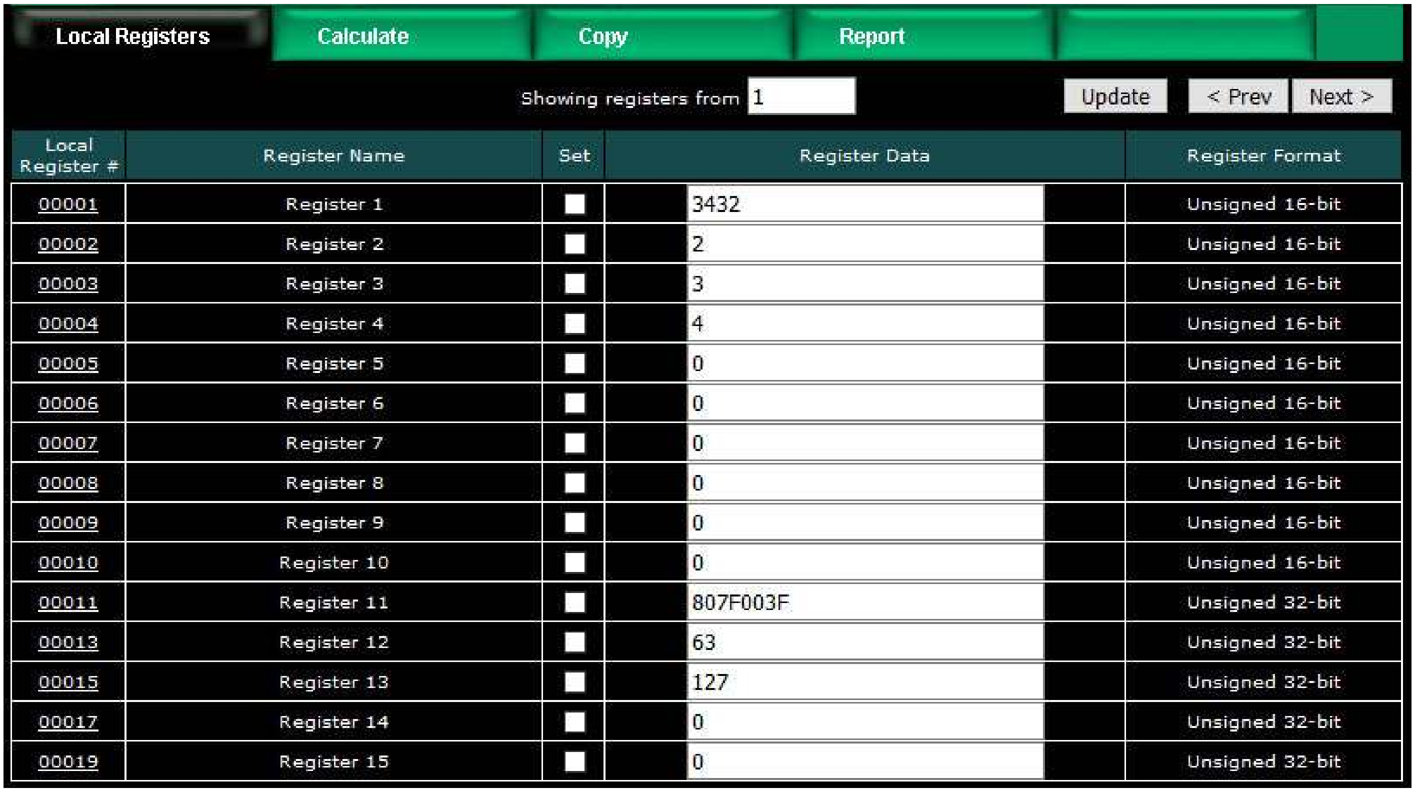


Showing 1 to 6 of 6

Rule #	Perform Operation	Using Register #	And/Thru Using	This Reg#/Value	Place Result in Register #
1	unpack	1	using	Fh	2
2	unpack	1	using	F0h	3
3	unpack	1	using	F00h	4
4	unpack	11	using	FFh	13
5	unpack	11	using	FF0000h	15
6	none	0	and	0	0

Rules Enabled: 6

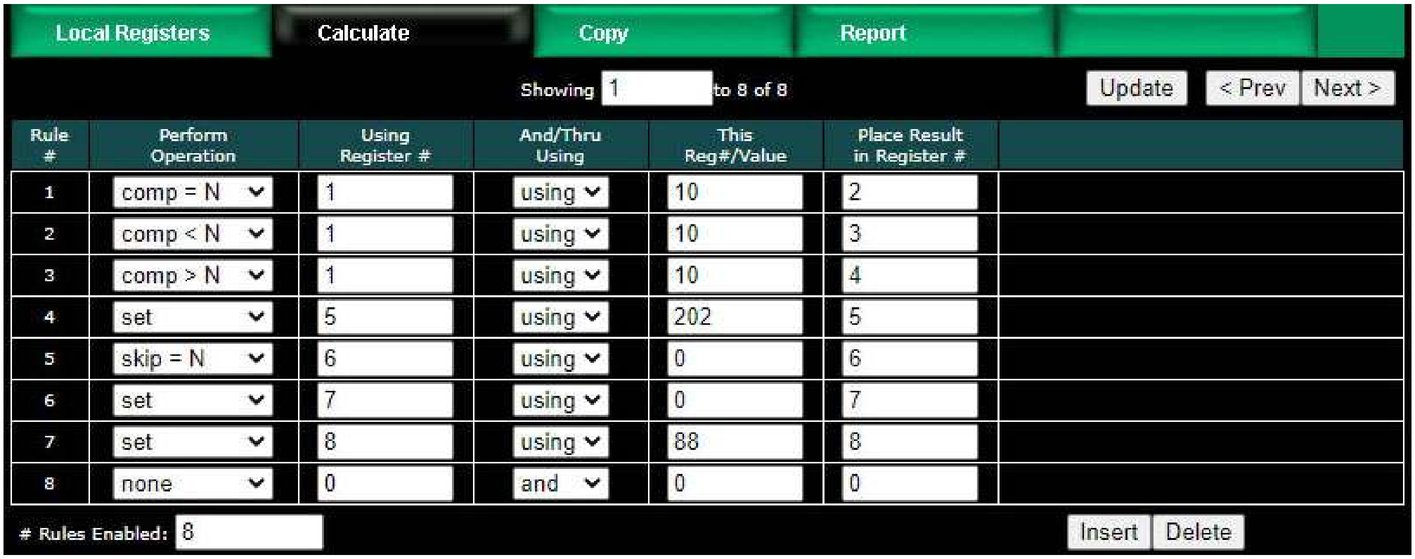
Register 1 is unpacked into registers 2, 3 and 4. Register 11 is unpacked into registers 13 and 15 (all 32-bit register pairs).



Showing registers from 1

Local Register #	Register Name	Set	Register Data	Register Format
00001	Register 1	□	3432	Unsigned 16-bit
00002	Register 2	□	2	Unsigned 16-bit
00003	Register 3	□	3	Unsigned 16-bit
00004	Register 4	□	4	Unsigned 16-bit
00005	Register 5	□	0	Unsigned 16-bit
00006	Register 6	□	0	Unsigned 16-bit
00007	Register 7	□	0	Unsigned 16-bit
00008	Register 8	□	0	Unsigned 16-bit
00009	Register 9	□	0	Unsigned 16-bit
00010	Register 10	□	0	Unsigned 16-bit
00011	Register 11	□	807F003F	Unsigned 32-bit
00013	Register 12	□	63	Unsigned 32-bit
00015	Register 13	□	127	Unsigned 32-bit
00017	Register 14	□	0	Unsigned 32-bit
00019	Register 15	□	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 6 contains a zero. If register 6 is not equal to zero, then rule 6 will be executed. (The numbers rule 6 and register 6 are not related in any other way, this is just conincidence in the example.)

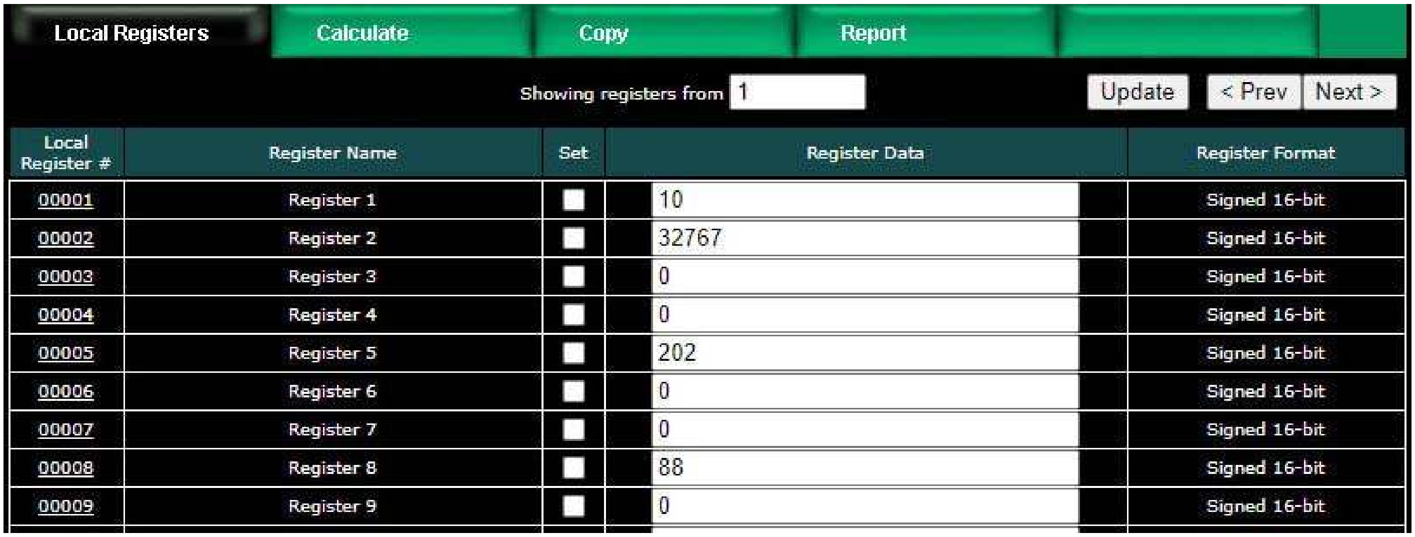


Showing 1 to 8 of 8

Rule #	Perform Operation	Using Register #	And/Thru Using	This Reg#/Value	Place Result in Register #
1	comp = N	1	using	10	2
2	comp < N	1	using	10	3
3	comp > N	1	using	10	4
4	set	5	using	202	5
5	skip = N	6	using	0	6
6	set	7	using	0	7
7	set	8	using	88	8
8	none	0	and	0	0

Rules Enabled: 8

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

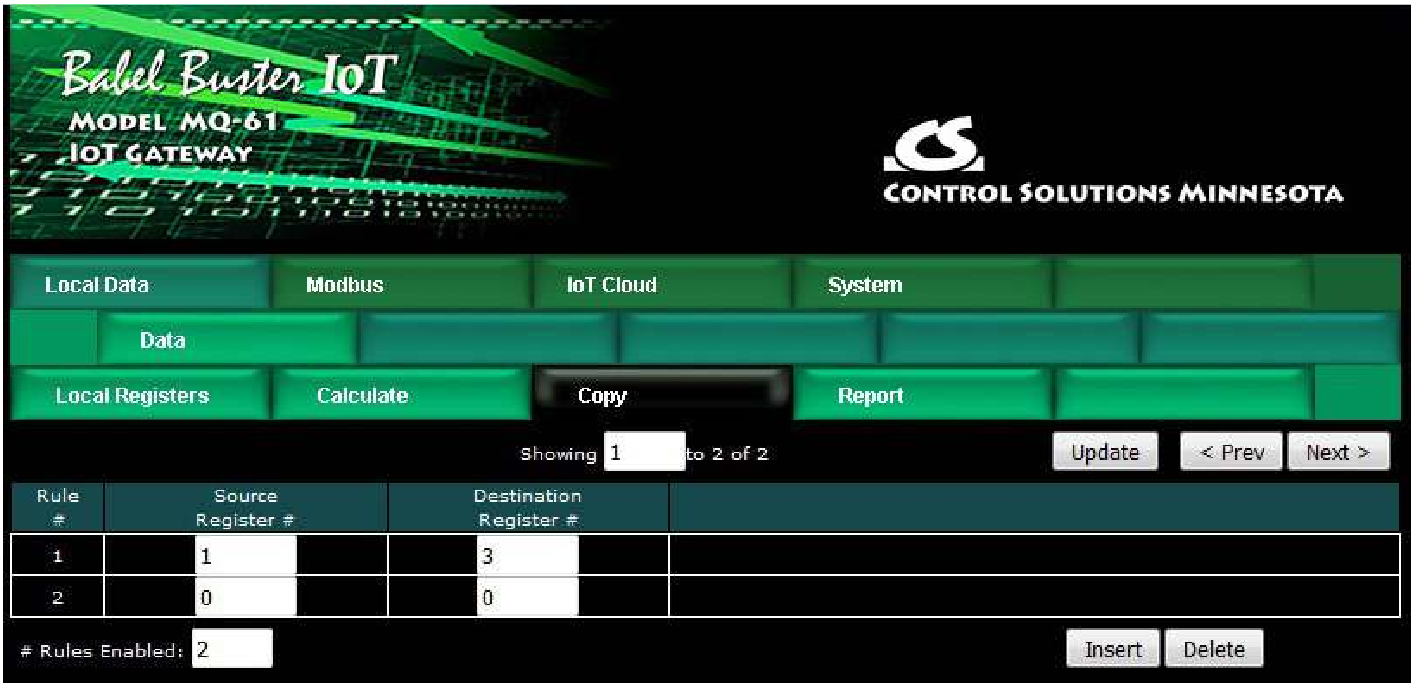


Showing registers from 1

Local Register #	Register Name	Set	Register Data	Register Format
00001	Register 1	□	10	Signed 16-bit
00002	Register 2	□	32767	Signed 16-bit
00003	Register 3	□	0	Signed 16-bit
00004	Register 4	□	0	Signed 16-bit
00005	Register 5	□	202	Signed 16-bit
00006	Register 6	□	0	Signed 16-bit
00007	Register 7	□	0	Signed 16-bit
00008	Register 8	□	88	Signed 16-bit
00009	Register 9	□	0	Signed 16-bit

4.4 Local Register Copy Rules

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

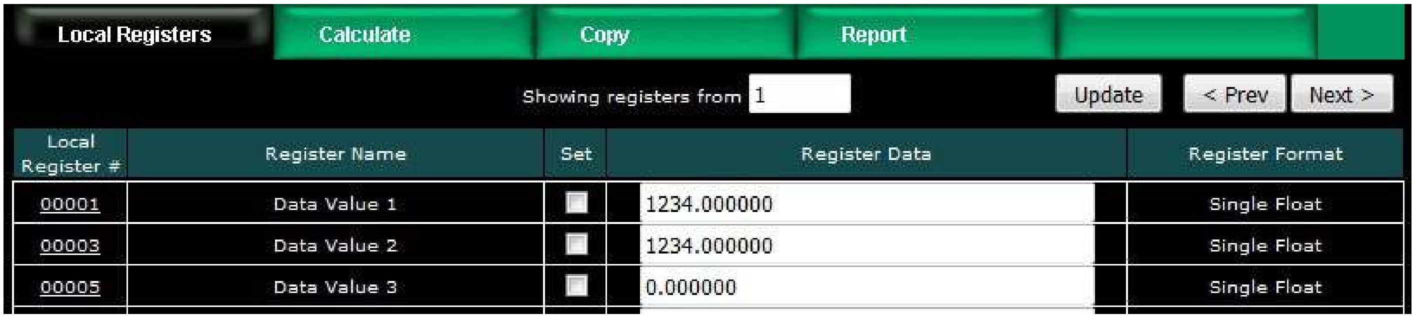


Showing 1 to 2 of 2

Rule #	Source Register #	Destination Register #
1	1	3
2	0	0

Rules Enabled: 2

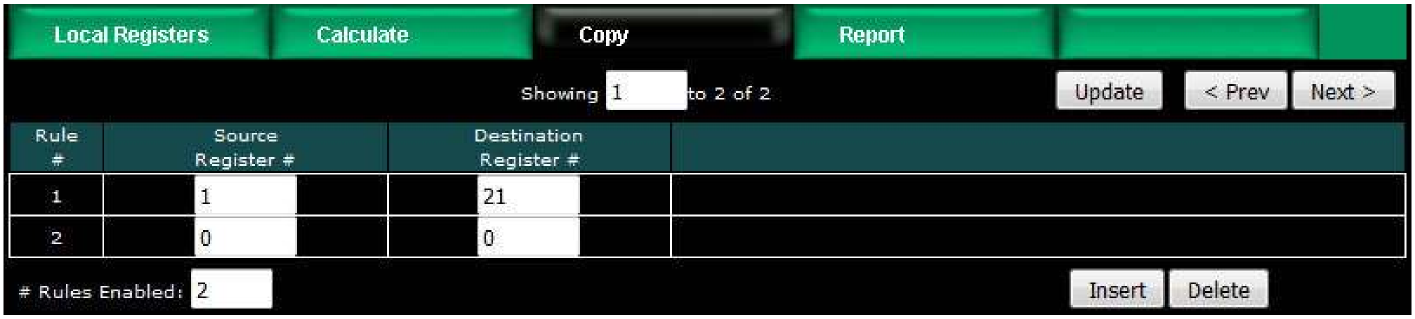
The above rule would cause the following data copy to occur:



Showing registers from 1

Local Register #	Register Name	Set	Register Data	Register Format
00001	Data Value 1	□	1234.000000	Single Float
00003	Data Value 2	□	1234.000000	Single Float
00005	Data Value 3	□	0.000000	Single Float

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

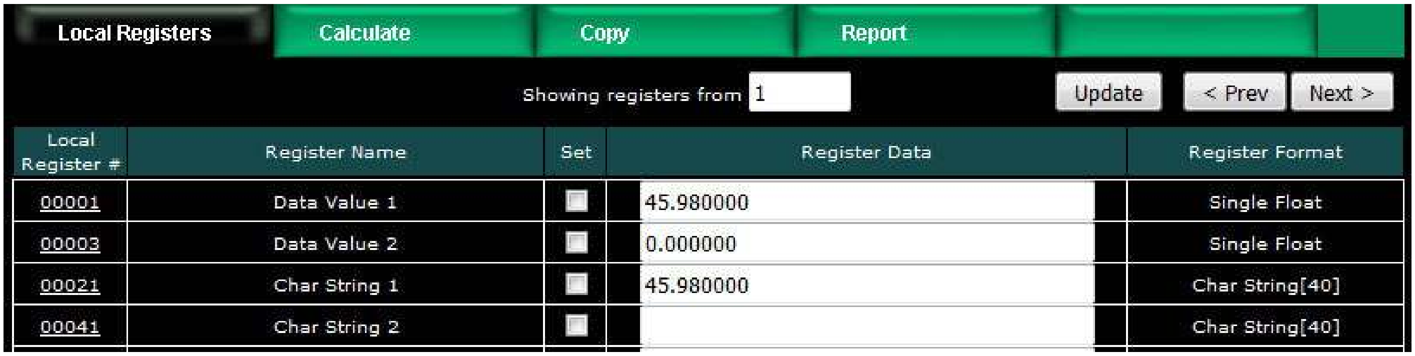


Showing 1 to 2 of 2

Rule #	Source Register #	Destination Register #
1	1	21
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.



Showing registers from 1

Local Register #	Register Name	Set	Register Data	Register Format
00001	Data Value 1	□	45.980000	Single Float
00003	Data Value 2	□	0.000000	Single Float
00021	Char String 1	□	45.980000	Char String[40]
00041	Char String 2	□		Char String[40]

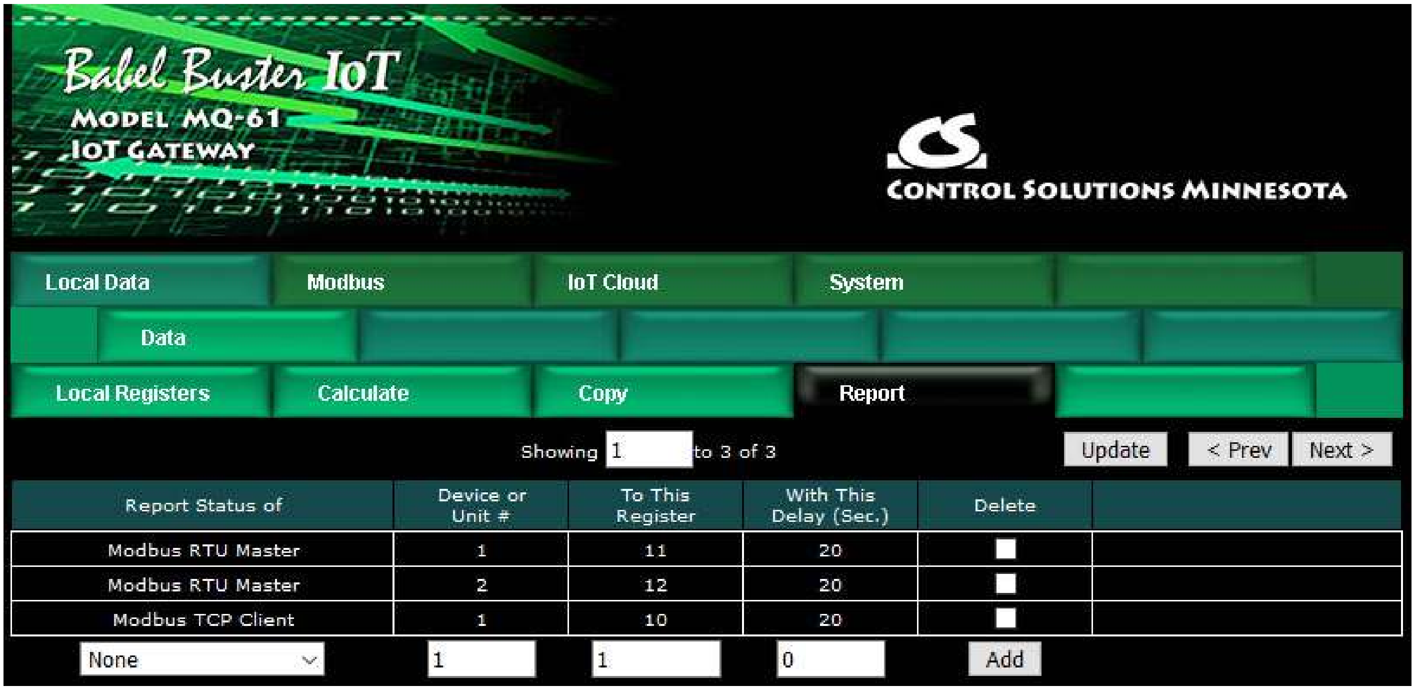
String conversion is not the only conversion you can do. If you need to convert floating point to integer, or vice versa, the copy rule will also do that. Note, however, that if you need to read an integer from a remote Modbus slave but want the result stored locally as floating point, you can do that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

4.5 Device Status Reporting

The Babel Buster IoT Gateway read maps include the ability to set a default value upon 'n' read fails, meaning that if the Babel Buster 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 IoT Gateway 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.



Report Status of	Device or Unit #	To This Register	With This Delay (Sec.)	Delete	
Modbus RTU Master	1	11	20	☐	
Modbus RTU Master	2	12	20	☐	
Modbus TCP Client	1	10	20	☐	
None	1	1	0	Add	

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.

Babel Buster IoT
MODEL MQ-61
MODEL BB3-6101-MQ
IOT GATEWAY

CONTROL SOLUTIONS MINNESOTA

5. Configuring Gateway as a Modbus RTU Master

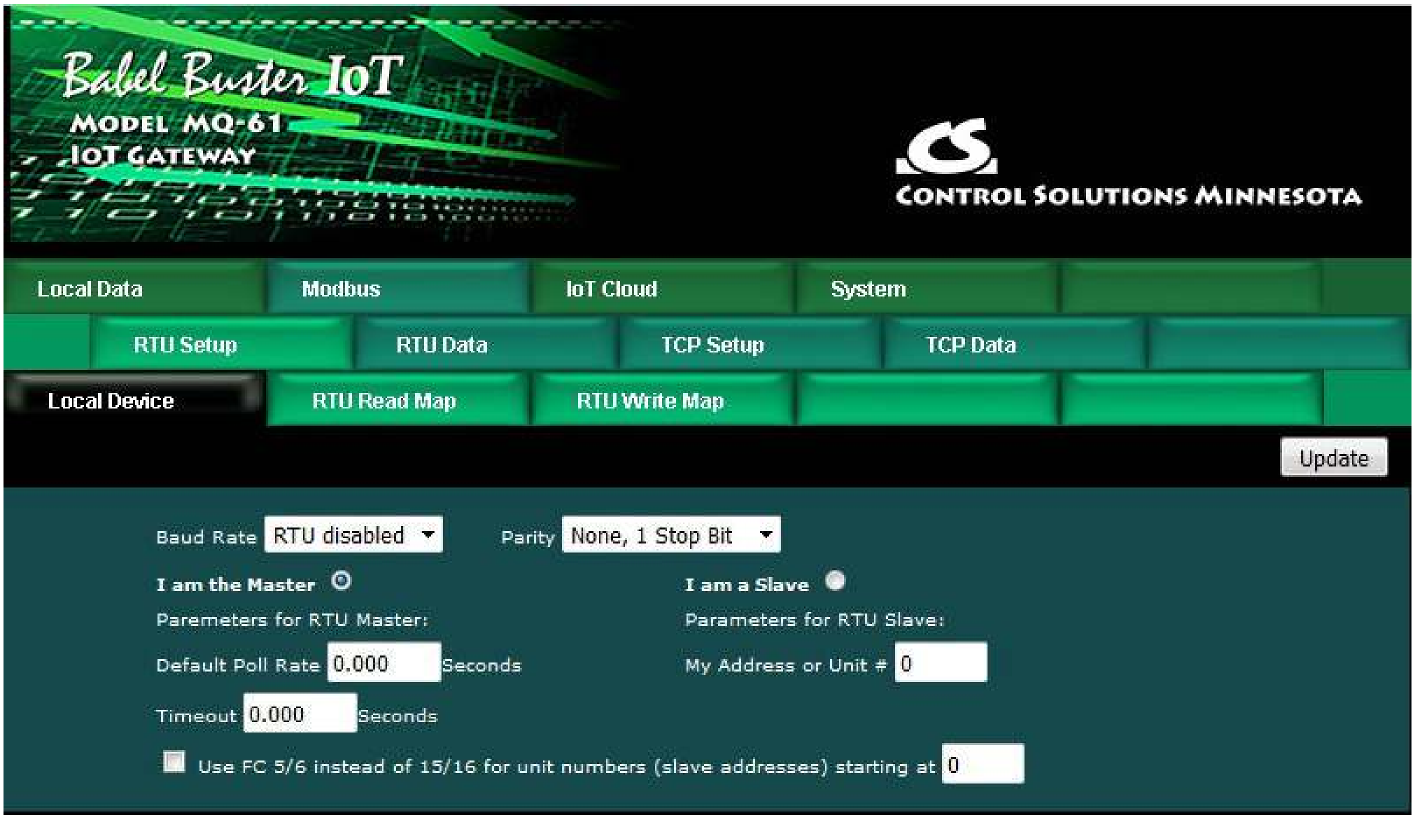
The Babel Buster IoT Gateway can be a Modbus RTU master or slave. As a master you can read Modbus data from, or write Modbus data to, other Modbus slaves. The gateway will periodically poll the other Modbus devices according to register maps you set up. To read from a remote Modbus device, configure 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 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.



Baud Rate RTU disabled Parity None, 1 Stop Bit

I am the Master ● I am a Slave ○

Paremeters for RTU Master: Parameters for RTU Slave:

Default Poll Rate 0.000 Seconds My Address or Unit # 0

Timeout 0.000 Seconds

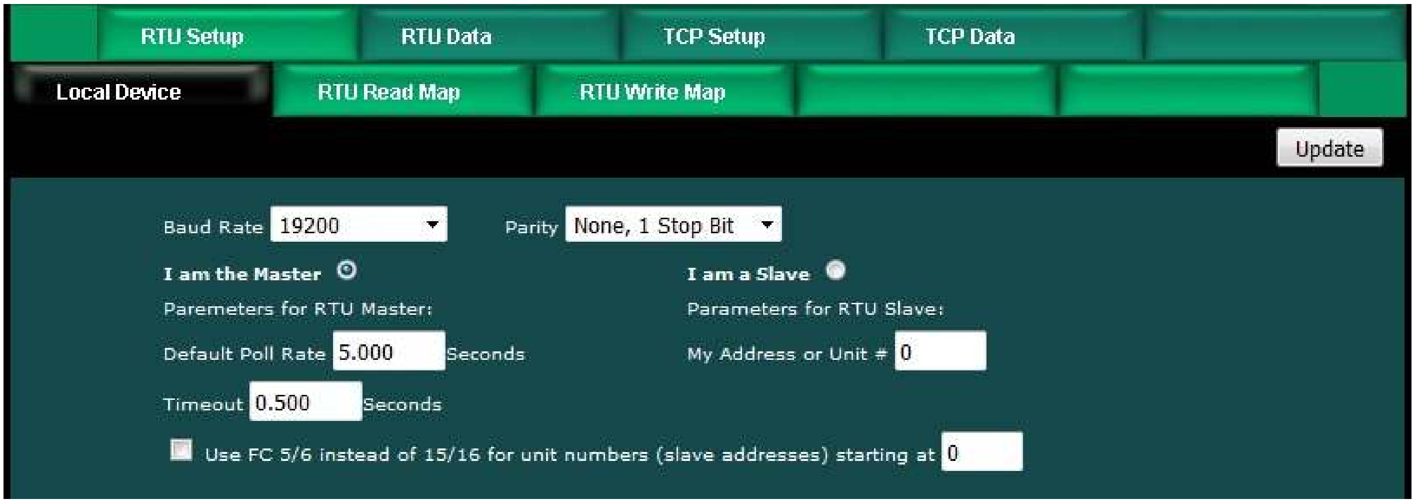
□ Use FC 5/6 instead of 15/16 for unit numbers (slave addresses) starting at 0

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 gateway 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 gateway is the slave.)



Baud Rate 19200 ▾ Parity None, 1 Stop Bit ▾

I am the Master ● I am a Slave ○

Paremeters for RTU Master: Parameters for RTU Slave:

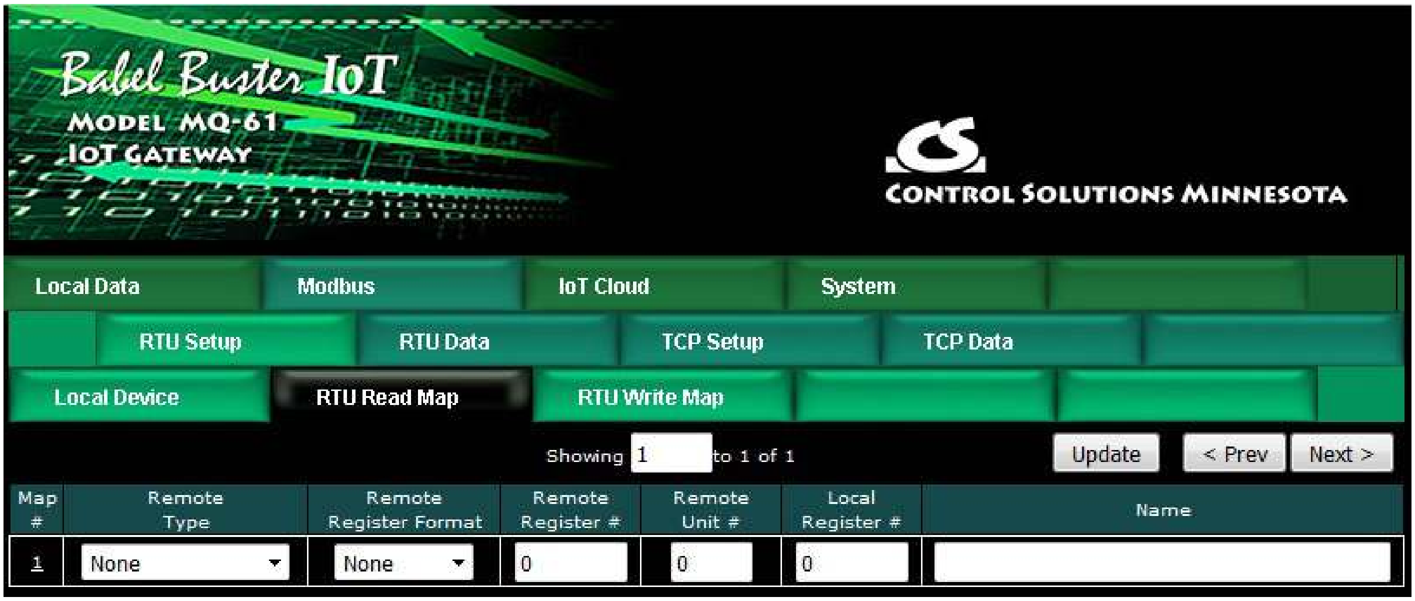
Default Poll Rate 5.000 Seconds My Address or Unit # 0

Timeout 0.500 Seconds

□ Use FC 5/6 instead of 15/16 for unit numbers (slave addresses) starting at 0

5.2 Modbus RTU Master Read Maps

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

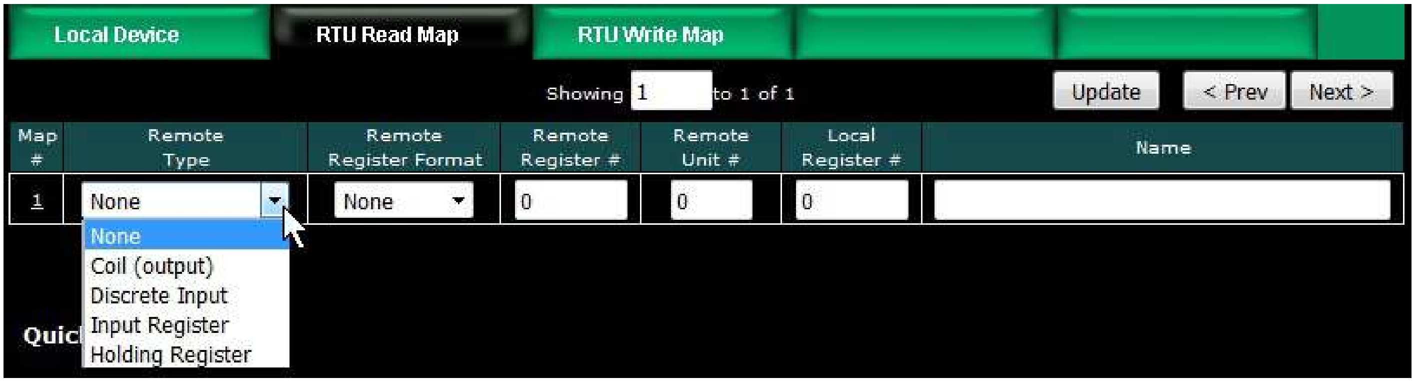


Showing 1 to 1 of 1

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.

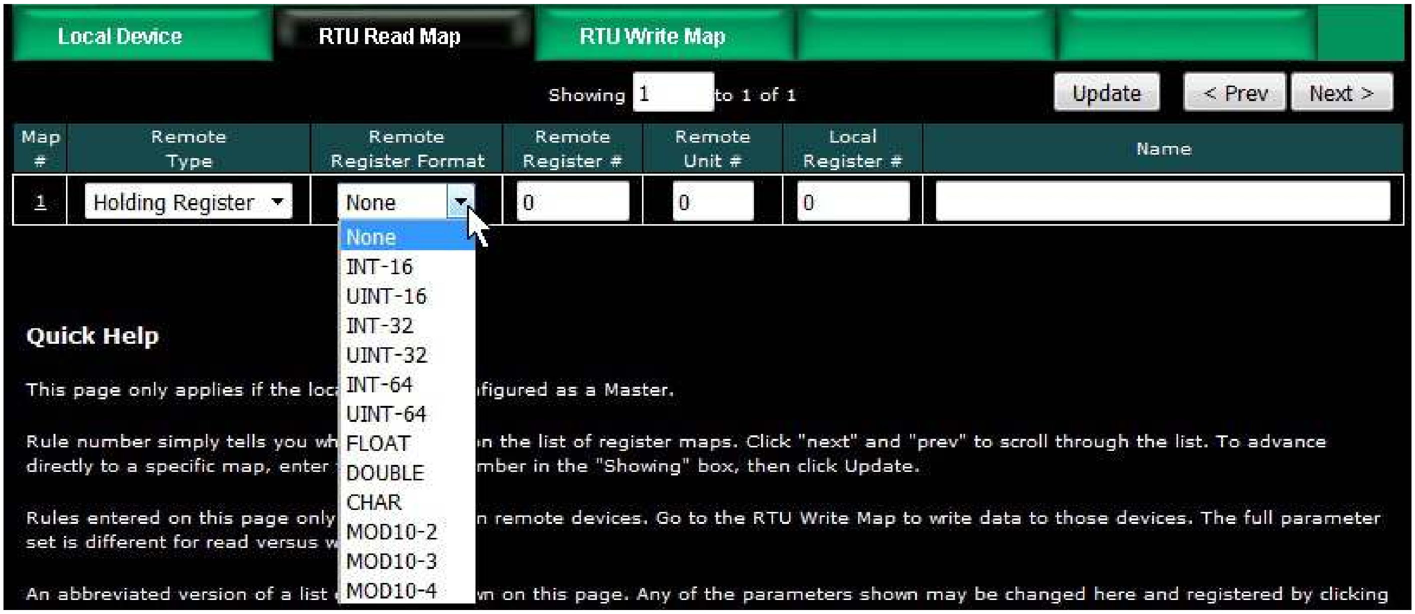
Maps entered on this page only read data from remote devices. Go to the RTU Write Map to write data to those devices. The full parameter set is different for read versus write.



Showing 1 to 1 of 1

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

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



Showing 1 to 1 of 1

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

Quick Help

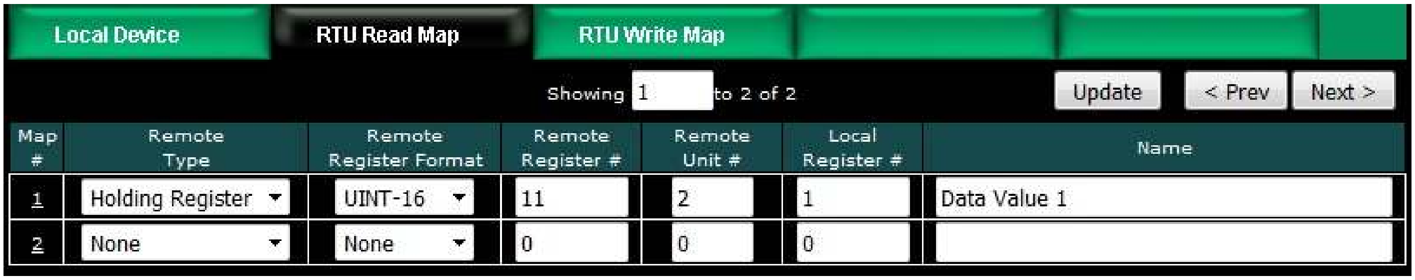
This page only applies if the loc ifigured as a Master.

Rule number simply tells you wh in the list of register maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter mber in the "Showing" box, then click Update.

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

An abbreviated version of a list wn on this page. Any of the parameters shown may be changed here and registered by clicking

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.)

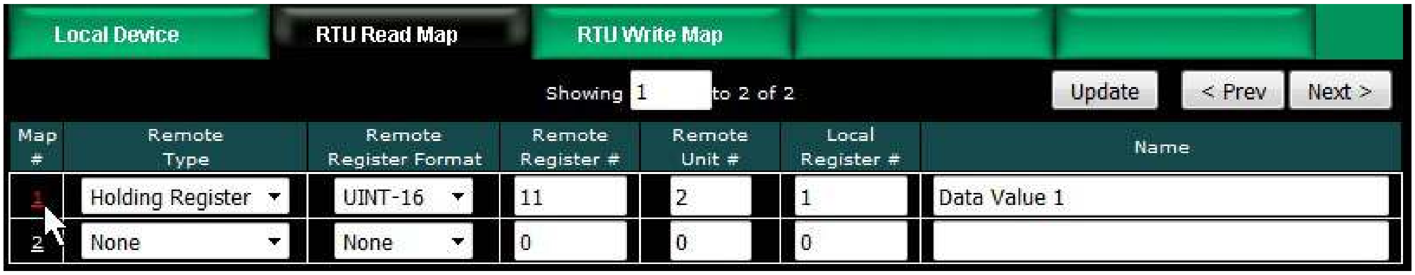


Showing 1 to 2 of 2

Map #	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Local Register #	Name
1	Holding Register	UINT-16	11	2	1	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.

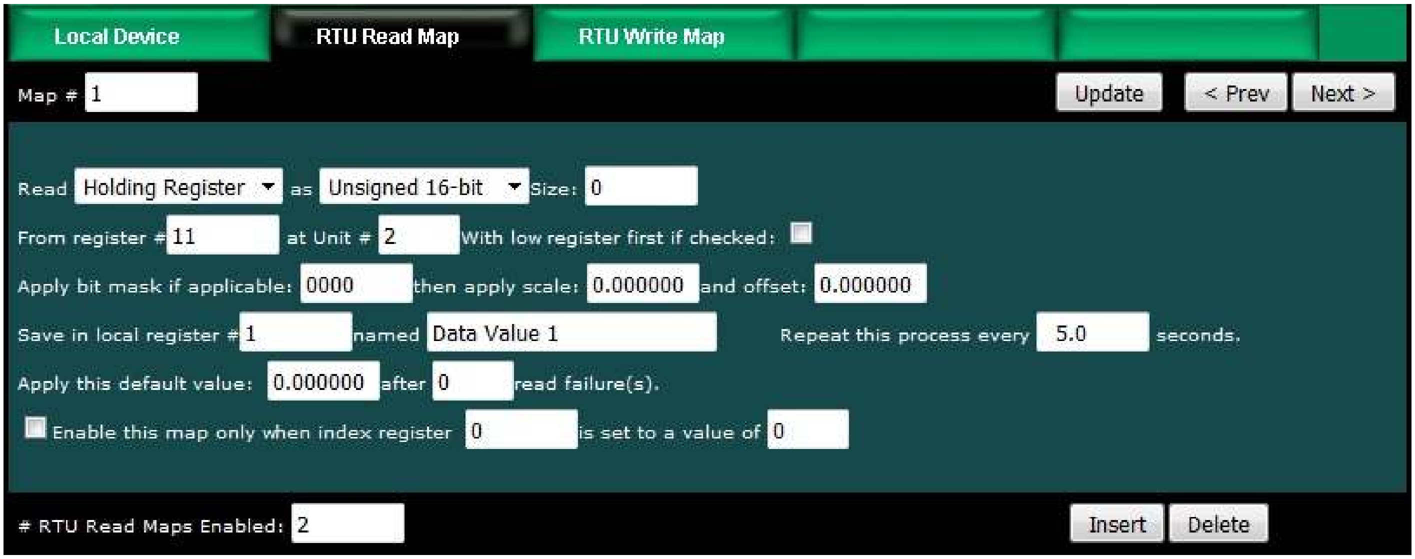
The Local Register is where data read from the remote Modbus RTU slave will be stored locally in the Babel Buster IoT Gateway. If the local register data format does not match what you are reading from the Modbus slave, the data will be converted automatically when it is read.



Showing 1 to 2 of 2

Map #	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Local Register #	Name
1	Holding Register	UINT-16	11	2	1	Data Value 1
2	None	None	0	0	0	

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



Map # 1

Read Holding Register as Unsigned 16-bit Size: 0

From register # 11 at Unit # 2 With low register first if checked: □

Apply bit mask if applicable: 0000 then apply scale: 0.000000 and offset: 0.000000

Save in local register # 1 named Data Value 1 Repeat this process every 5.0 seconds.

Apply this default value: 0.000000 after 0 read failure(s).

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

RTU Read Maps Enabled: 2

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, Babel Buster 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.

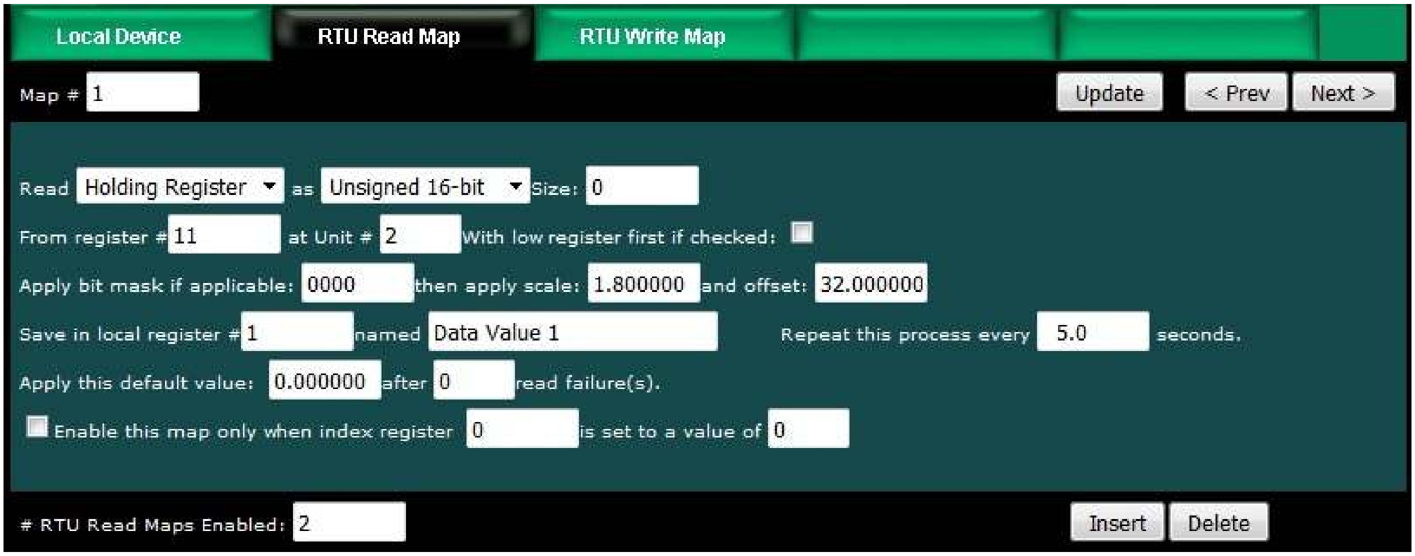
You have the option of making this Read Map conditional. If an index register number is provided and the Enable box is checked, then this read map will only be executed when the index register (local register) contains the value given. This allows multiple read maps to supply data to the same local register 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.



Map # 1

Read Holding Register as Unsigned 16-bit Size: 0

From register # 11 at Unit # 2 With low register first if checked: □

Apply bit mask if applicable: 0000 then apply scale: 1.800000 and offset: 32.000000

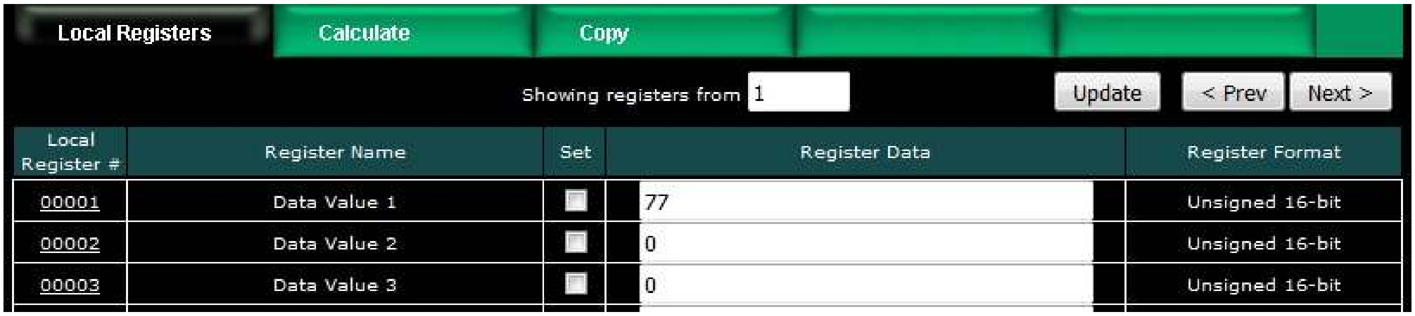
Save in local register # 1 named Data Value 1 Repeat this process every 5.0 seconds.

Apply this default value: 0.000000 after 0 read failure(s).

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

RTU Read Maps Enabled: 2

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).

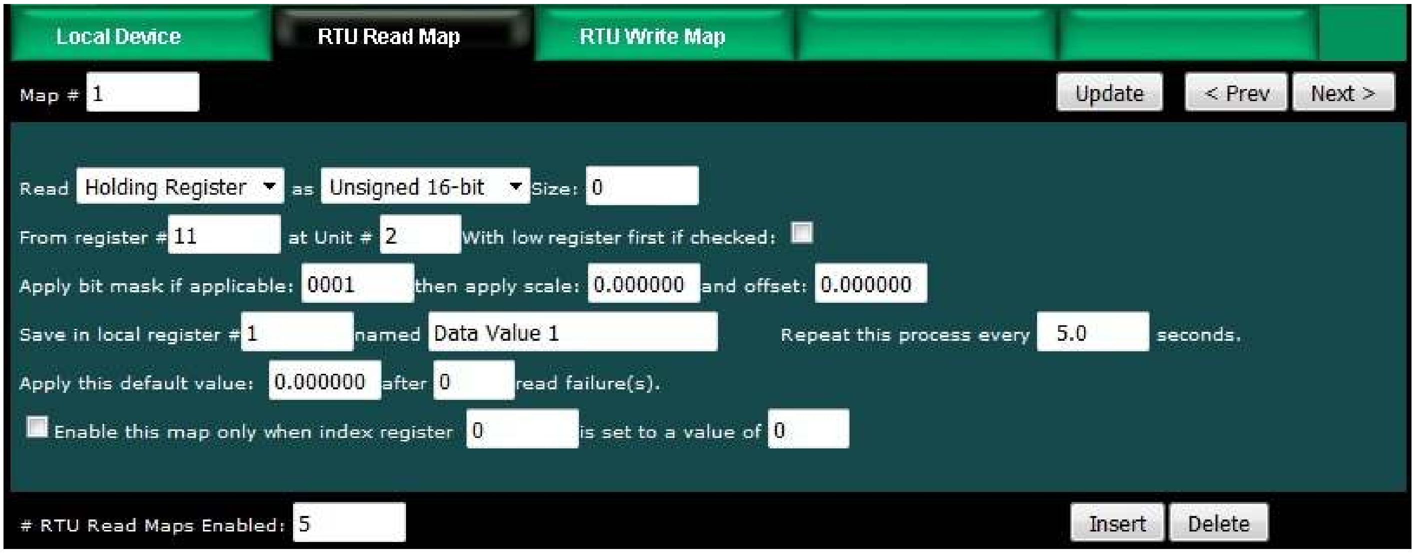


Showing registers from 1

Local Register #	Register Name	Set	Register Data	Register Format
00001	Data Value 1	□	77	Unsigned 16-bit
00002	Data Value 2	□	0	Unsigned 16-bit
00003	Data Value 3	□	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. Babel Buster 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.



Map # 1

Read Holding Register as Unsigned 16-bit Size: 0

From register # 11 at Unit # 2 With low register first if checked: □

Apply bit mask if applicable: 0001 then apply scale: 0.000000 and offset: 0.000000

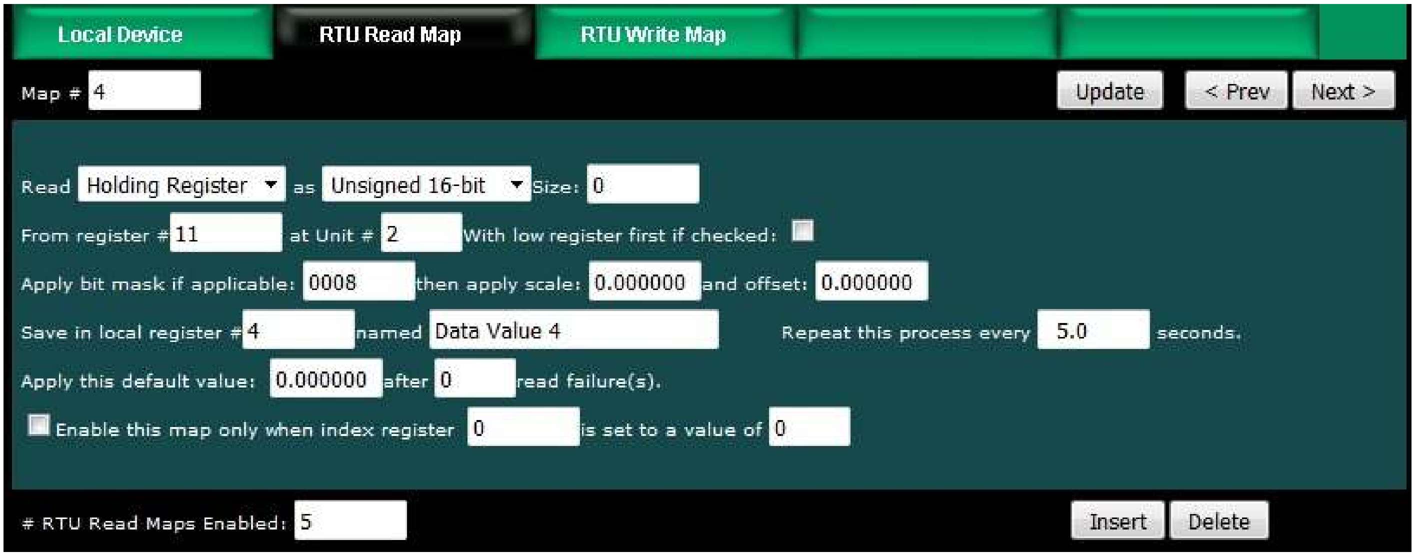
Save in local register # 1 named Data Value 1 Repeat this process every 5.0 seconds.

Apply this default value: 0.000000 after 0 read failure(s).

□ 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.



Map # 4

Read Holding Register as Unsigned 16-bit Size: 0

From register # 11 at Unit # 2 With low register first if checked: □

Apply bit mask if applicable: 0008 then apply scale: 0.000000 and offset: 0.000000

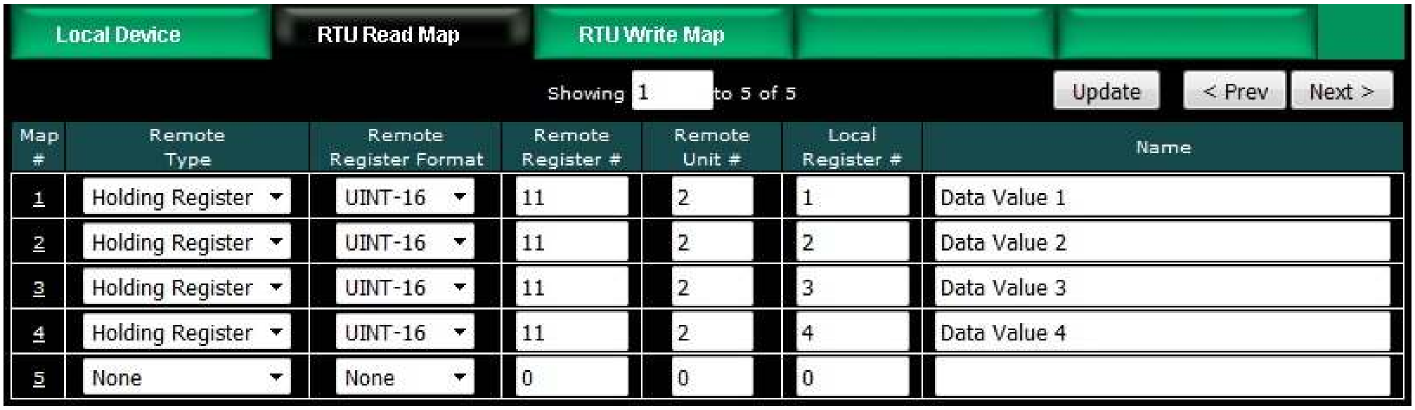
Save in local register # 4 named Data Value 4 Repeat this process every 5.0 seconds.

Apply this default value: 0.000000 after 0 read failure(s).

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

RTU Read Maps Enabled: 5

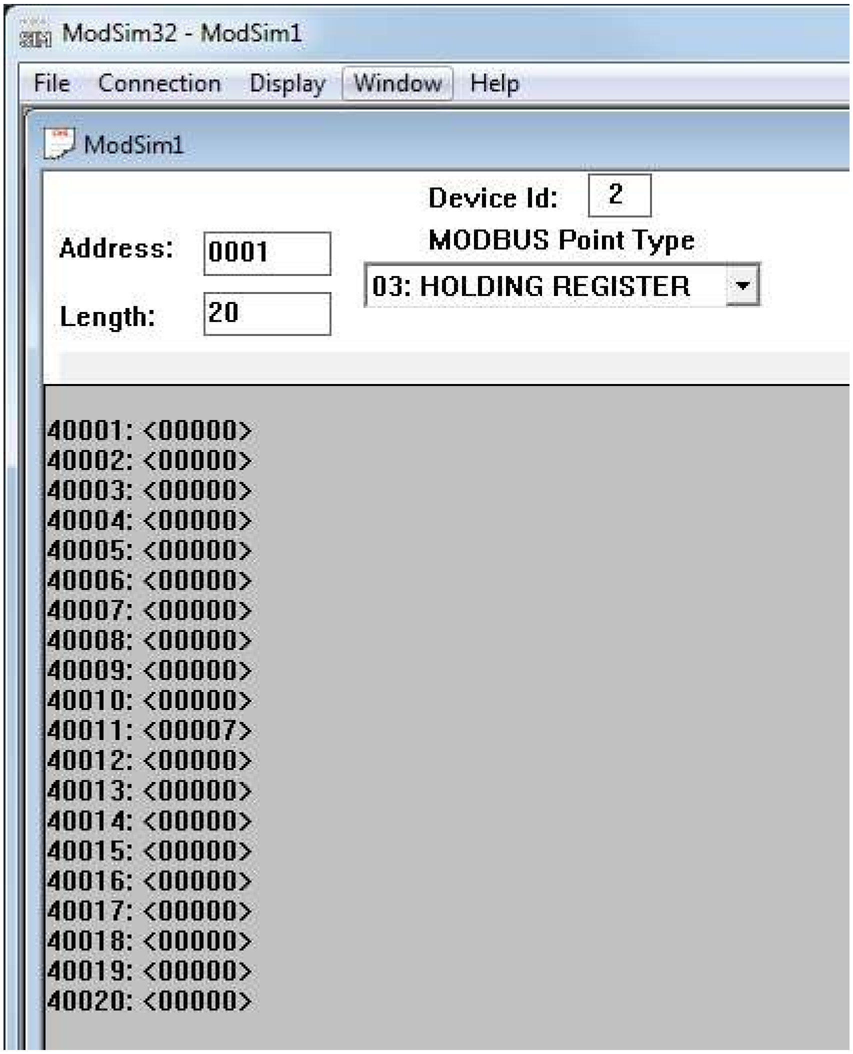
If the read maps referencing the same remote register are created in sequential contiguous order, the Babel Buster will optimize the RTU activity by reading the remote register once and then sharing the data with all of the read maps in the group. In the example illustrated here, four consecutive read 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.



Showing 1 to 5 of 5

Map #	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Local Register #	Name
1	Holding Register	UINT-16	11	2	1	Data Value 1
2	Holding Register	UINT-16	11	2	2	Data Value 2
3	Holding Register	UINT-16	11	2	3	Data Value 3
4	Holding Register	UINT-16	11	2	4	Data Value 4
5	None	None	0	0	0	

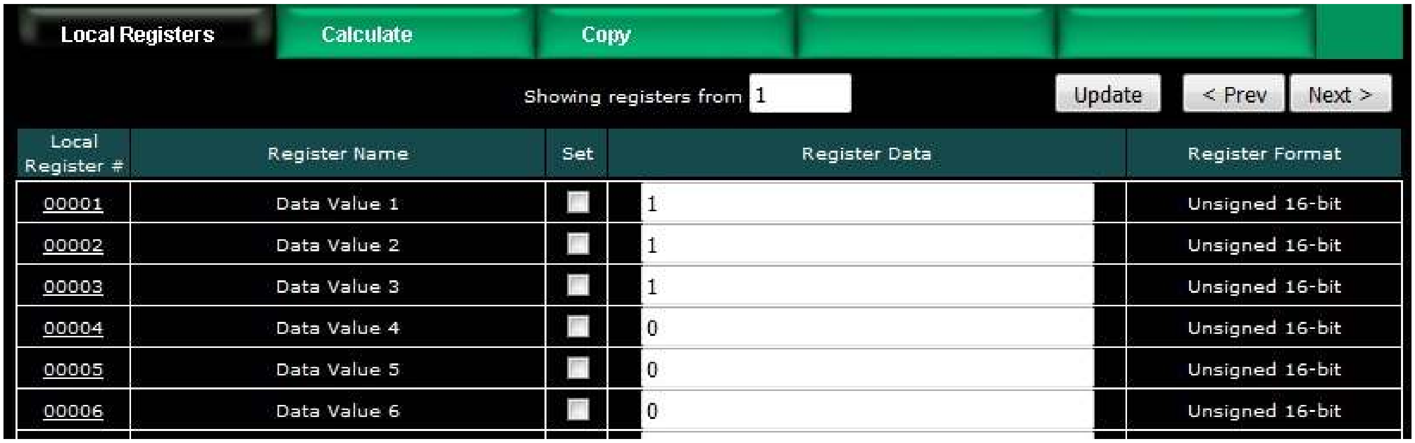
To try out these read maps, we have set up ModSim with the remote register 11 containing a value of 7.



Device Id: 2
Address: 0001
MODBUS Point Type: 03: HOLDING REGISTER
Length: 20

```
40001: <00000>
40002: <00000>
40003: <00000>
40004: <00000>
40005: <00000>
40006: <00000>
40007: <00000>
40008: <00000>
40009: <00000>
40010: <00000>
40011: <00007>
40012: <00000>
40013: <00000>
40014: <00000>
40015: <00000>
40016: <00000>
40017: <00000>
40018: <00000>
40019: <00000>
40020: <00000>
```

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

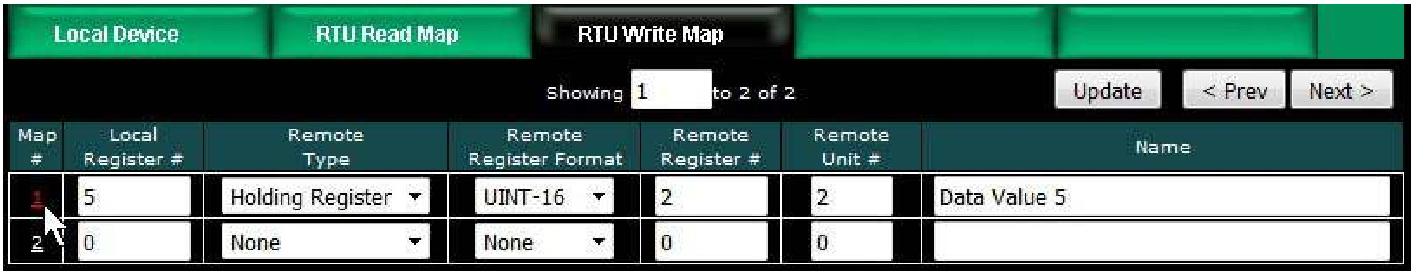


Showing registers from 1

Local Register #	Register Name	Set	Register Data	Register Format
00001	Data Value 1	□	1	Unsigned 16-bit
00002	Data Value 2	□	1	Unsigned 16-bit
00003	Data Value 3	□	1	Unsigned 16-bit
00004	Data Value 4	□	0	Unsigned 16-bit
00005	Data Value 5	□	0	Unsigned 16-bit
00006	Data Value 6	□	0	Unsigned 16-bit

5.3 Modbus RTU Master Write Maps

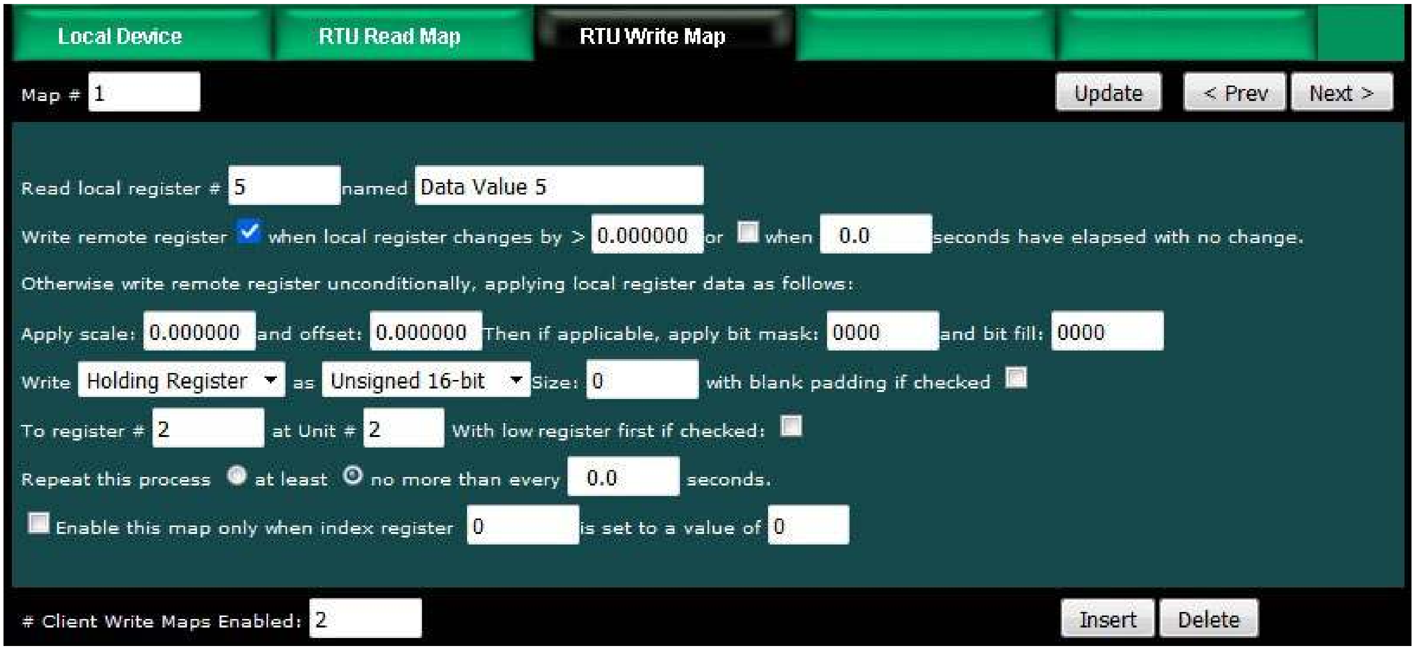
Getting the gateway to write registers to another Modbus device requires setting up a "Write Map" as shown here. Much of the Write Map is configured the same as a Read Map.



Showing 1 to 2 of 2

Map #	Local Register #	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Name
1	5	Holding Register	UINT-16	2	2	Data Value 5
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.



Map # 1

Read local register # 5 named Data Value 5

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 Unit # 2 With low register first if checked: □

Repeat this process ○ at least ● no more than every 0.0 seconds.

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

Client Write Maps Enabled: 2

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, Babel Buster 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.

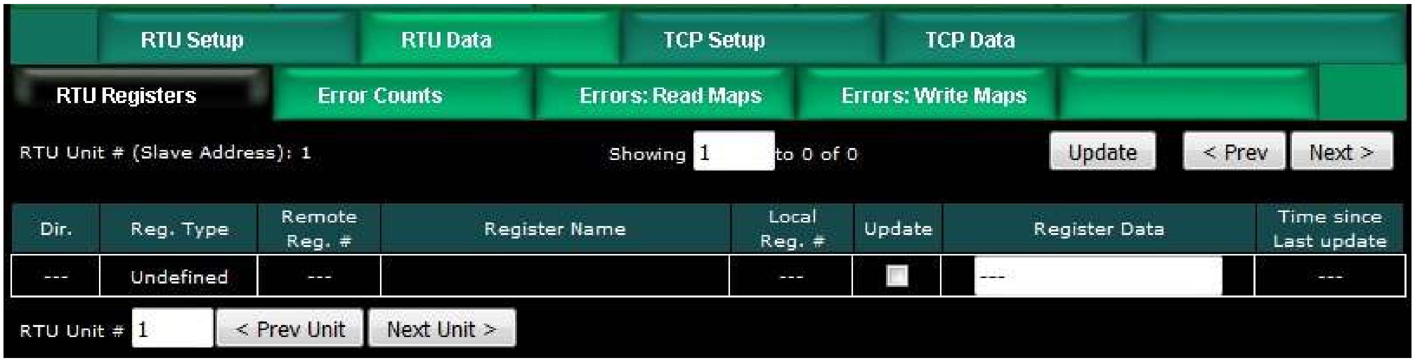
You have the option of making this Write Map conditional. If an index register number is provided and the Enable box is checked, then this write map will only be executed when the index register (local register) contains the value given. This allows multiple write maps to supply data to the same remote 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.



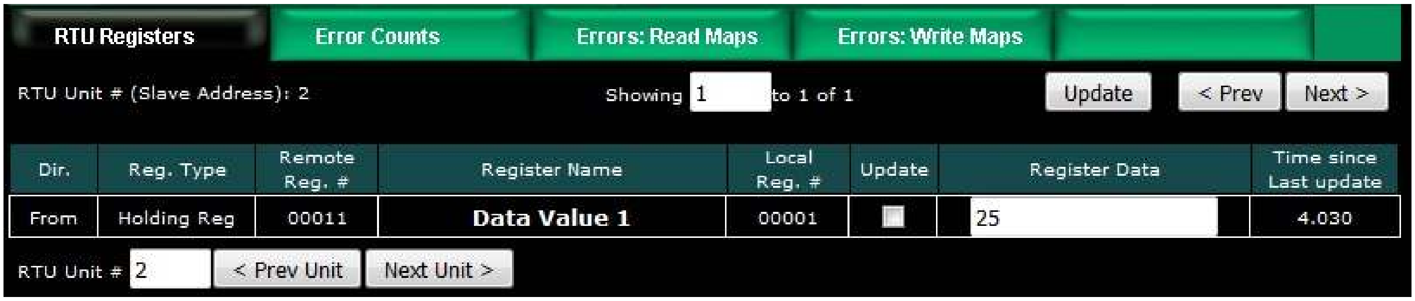
RTU Unit # (Slave Address): 1

Showing 1 to 0 of 0

Dir.	Reg. Type	Remote Reg. #	Register Name	Local Reg. #	Update	Register Data	Time since Last update
---	Undefined	---		---	□	---	---

RTU Unit # 1

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.



RTU Unit # (Slave Address): 2

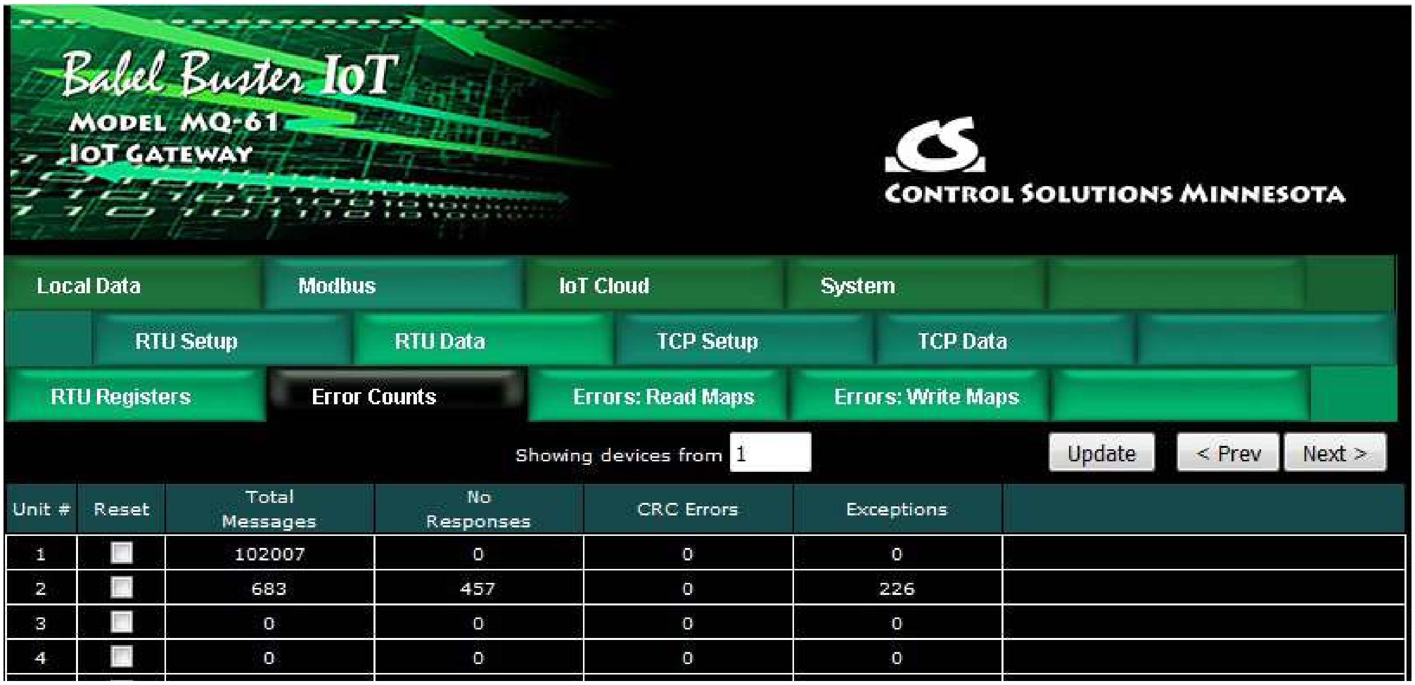
Showing 1 to 1 of 1

Dir.	Reg. Type	Remote Reg. #	Register Name	Local Reg. #	Update	Register Data	Time since Last update
From	Holding Reg	00011	**Data Value 1**	00001	□	25	4.030

RTU Unit # 2

5.5 Modbus RTU Errors

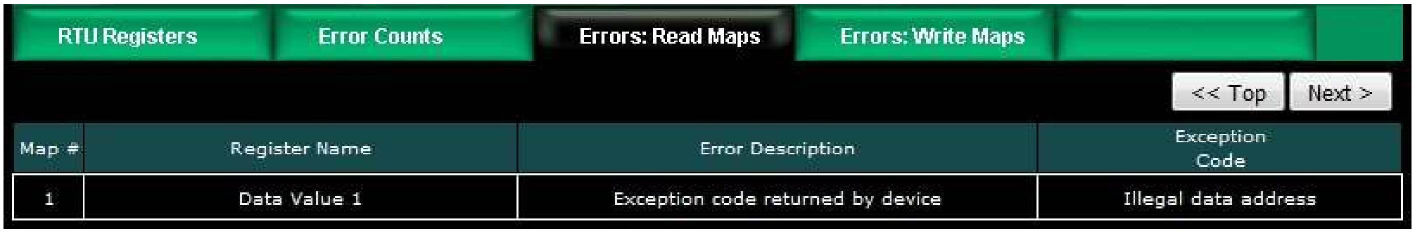
The Error Counts page shows a tabulation, by RTU slave, of all errors observed. In the example below, we can see that the RTU device at slave address 1 is running flawlessly while slave #2 has had some issues.



Showing devices from 1

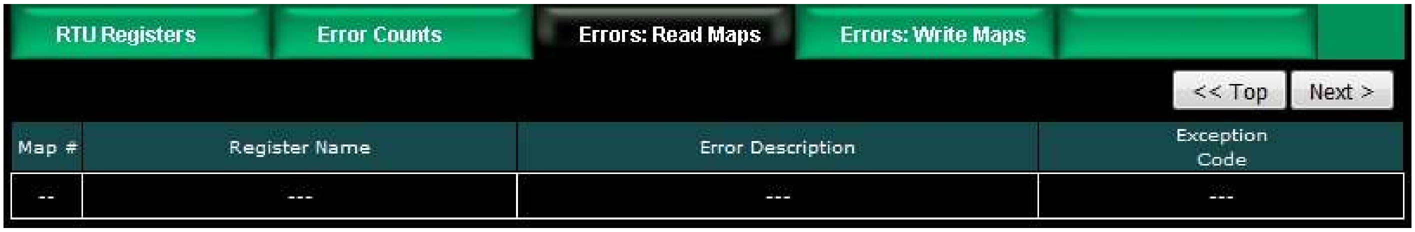
Unit #	Reset	Total Messages	No Responses	CRC Errors	Exceptions
1	□	102007	0	0	0
2	□	683	457	0	226
3	□	0	0	0	0
4	□	0	0	0	0

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.



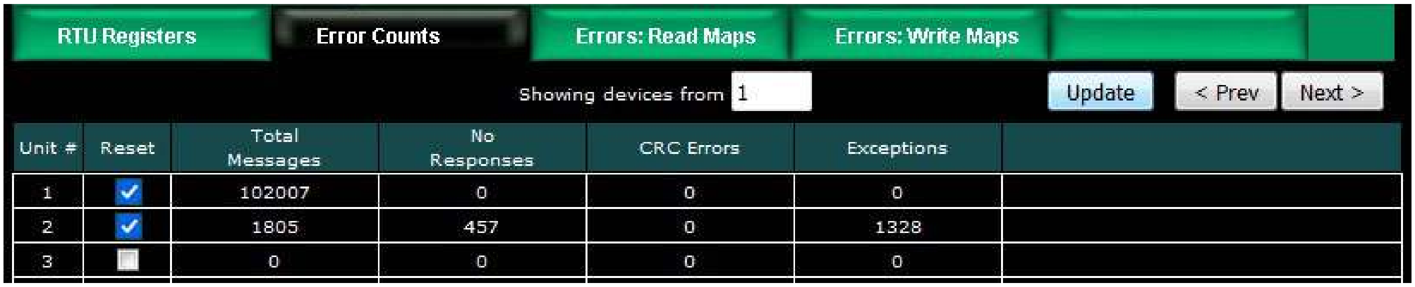
Map #	Register Name	Error Description	Exception Code
1	Data Value 1	Exception code returned by device	Illegal data address

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



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

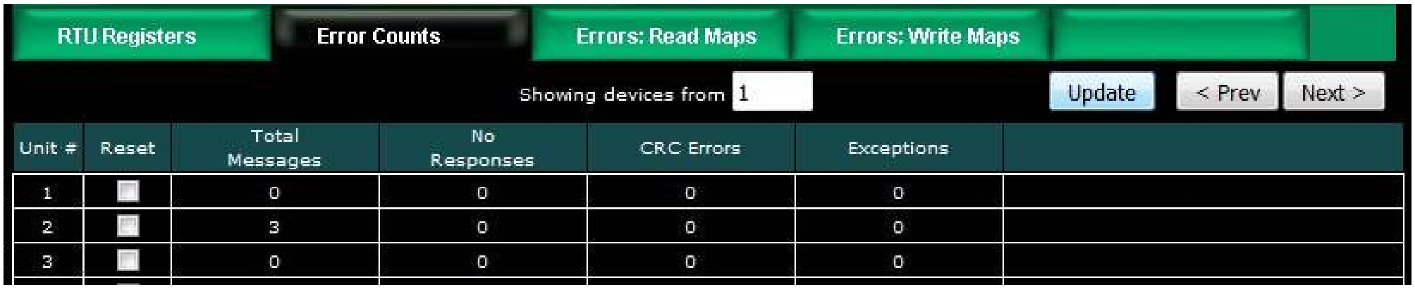
Once you have resolved problems, you can reset the error counts by checking the box in the Reset column and then clicking Update.



Showing devices from 1

Unit #	Reset	Total Messages	No Responses	CRC Errors	Exceptions
1	☑	102007	0	0	0
2	☑	1805	457	0	1328
3	□	0	0	0	0

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



Showing devices from 1

Unit #	Reset	Total Messages	No Responses	CRC Errors	Exceptions
1	□	0	0	0	0
2	□	3	0	0	0
3	□	0	0	0	0

Babel Buster IoT
MODEL MQ-61
MODEL BB3-6101-MQ
IOT GATEWAY

CONTROL SOLUTIONS MINNESOTA

6. Configuring Gateway as a Modbus TCP Client

The Babel Buster IoT Gateway 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 gateway 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.

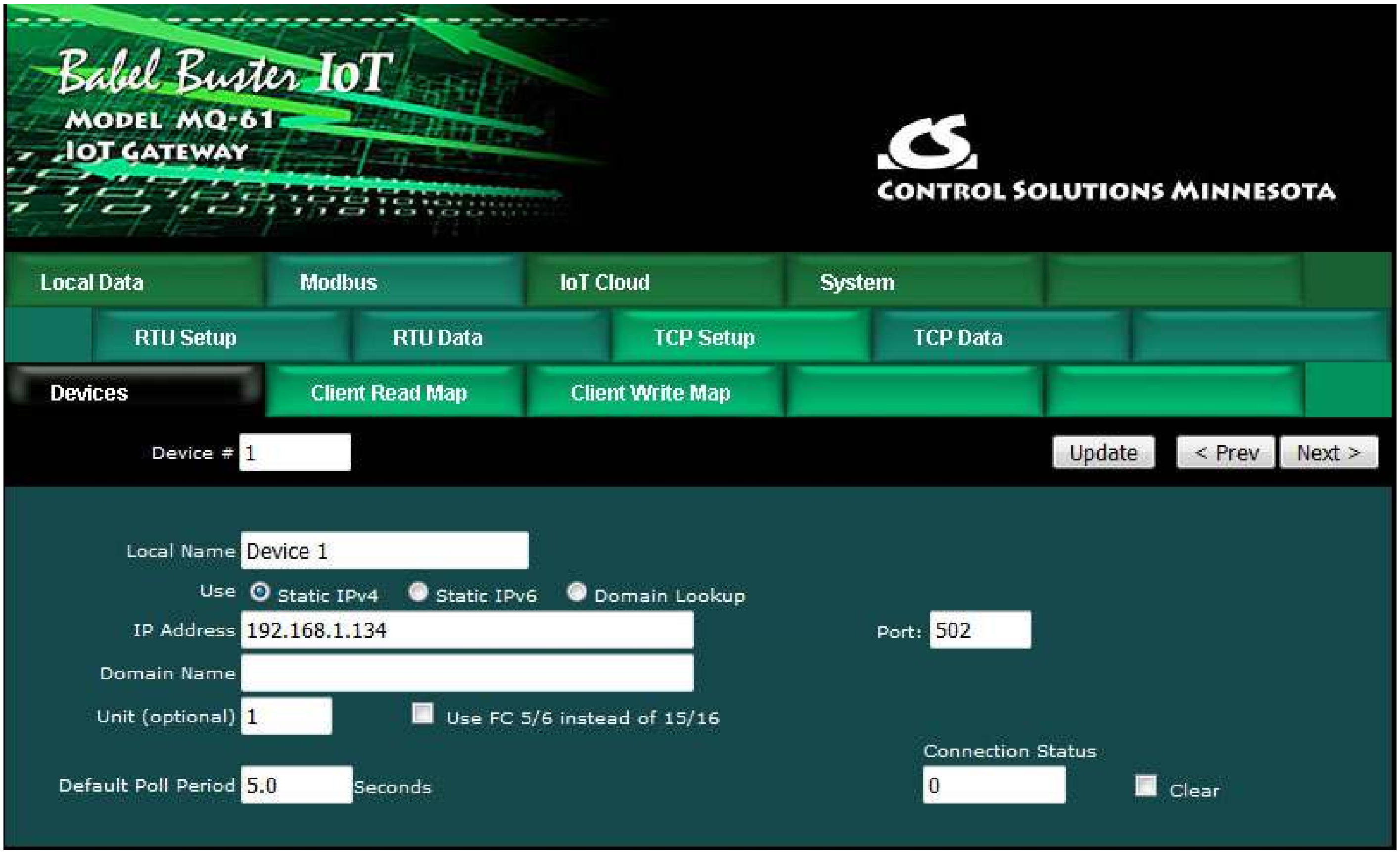
Data read from a remote device is stored in a local register when received. Data written to a remote device is taken from a local register when sent. The local registers are the same collection of registers that are accessible to other Modbus TCP clients as a collection of holding registers. These local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



Device # 1

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

Connection Status 0 □ Clear

Default Poll Period 5.0 Seconds

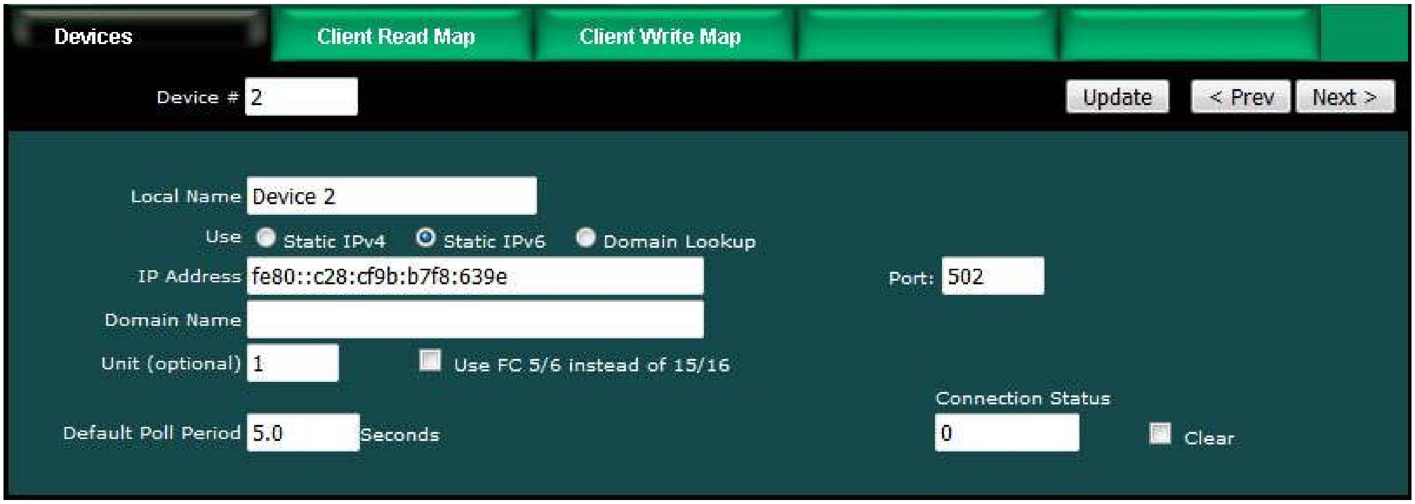
A unit number is always included in each Modbus TCP packet. This is the equivalent of the RTU slave address. Some TCP devices pay no attention to unit and simply echo back whatever you had sent. However, if you are accessing RTU devices on the other side of a TCP to RTU router (gateway), then the unit 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. (Control Solutions BB2-6010-GW or any other model with the -GW suffix would provide this type of TCP to RTU routing.)

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 gateway 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.



Device # 2

Local Name Device 2

Use ○ Static IPv4 ● Static IPv6 ○ Domain Lookup

IP Address fe80::c28:cf9b:b7f8:639e Port: 502

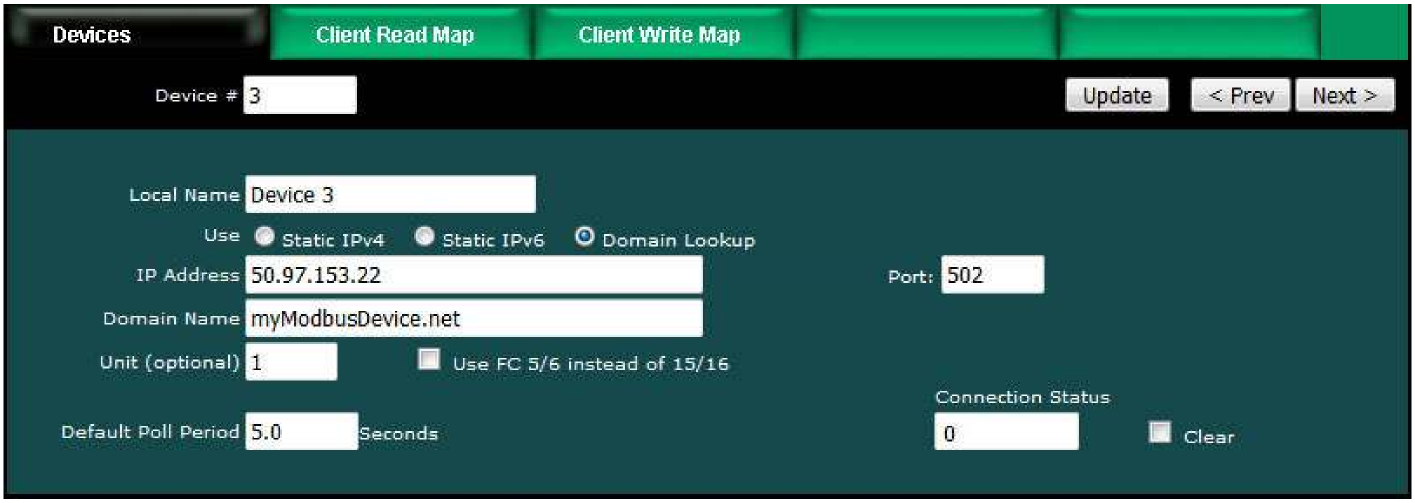
Domain Name

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

Default Poll Period 5.0 Seconds

Connection Status 0 □ Clear

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).



Device # 3

Local Name Device 3

Use ○ Static IPv4 ○ Static IPv6 ● Domain Lookup

IP Address 50.97.153.22 Port: 502

Domain Name myModbusDevice.net

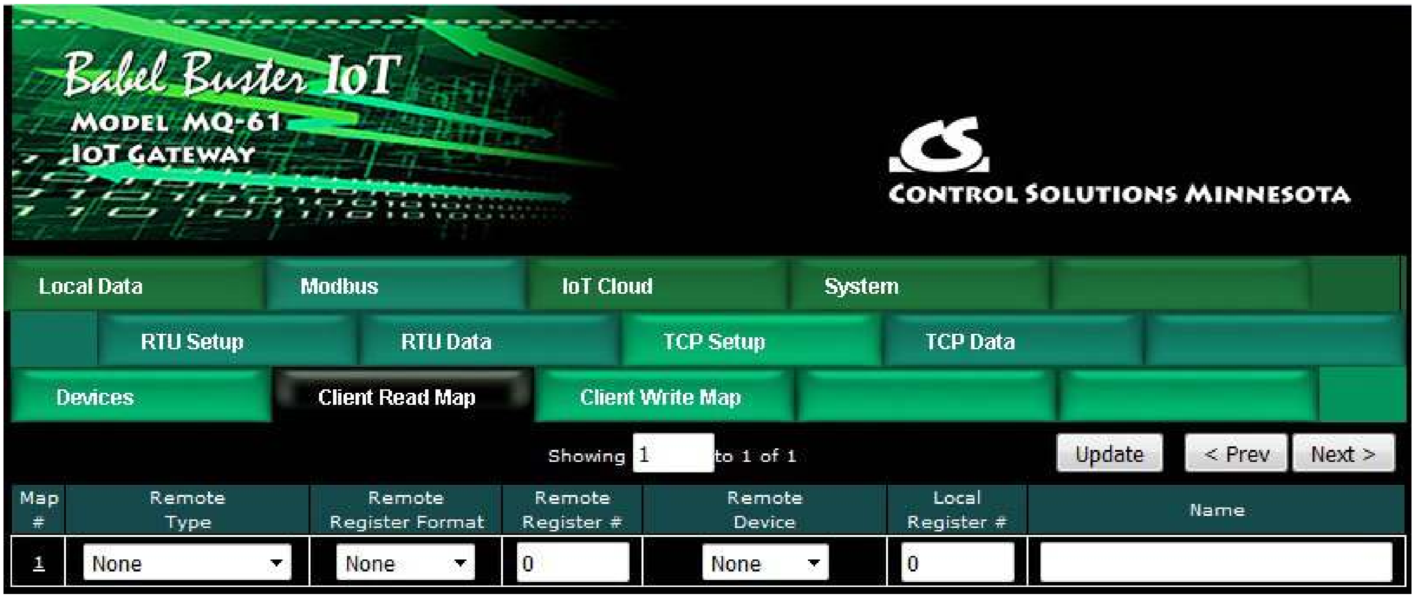
Unit (optional) 1 □ Use FC 5/6 instead of 15/16

Default Poll Period 5.0 Seconds

Connection Status 0 □ Clear

6.2 Modbus TCP Client Read Maps

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

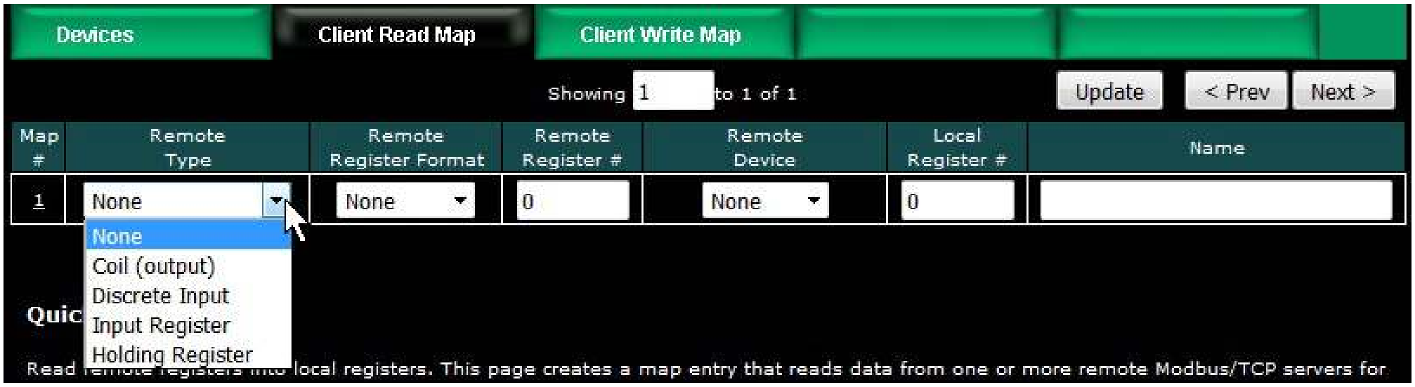


Showing 1 to 1 of 1

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.



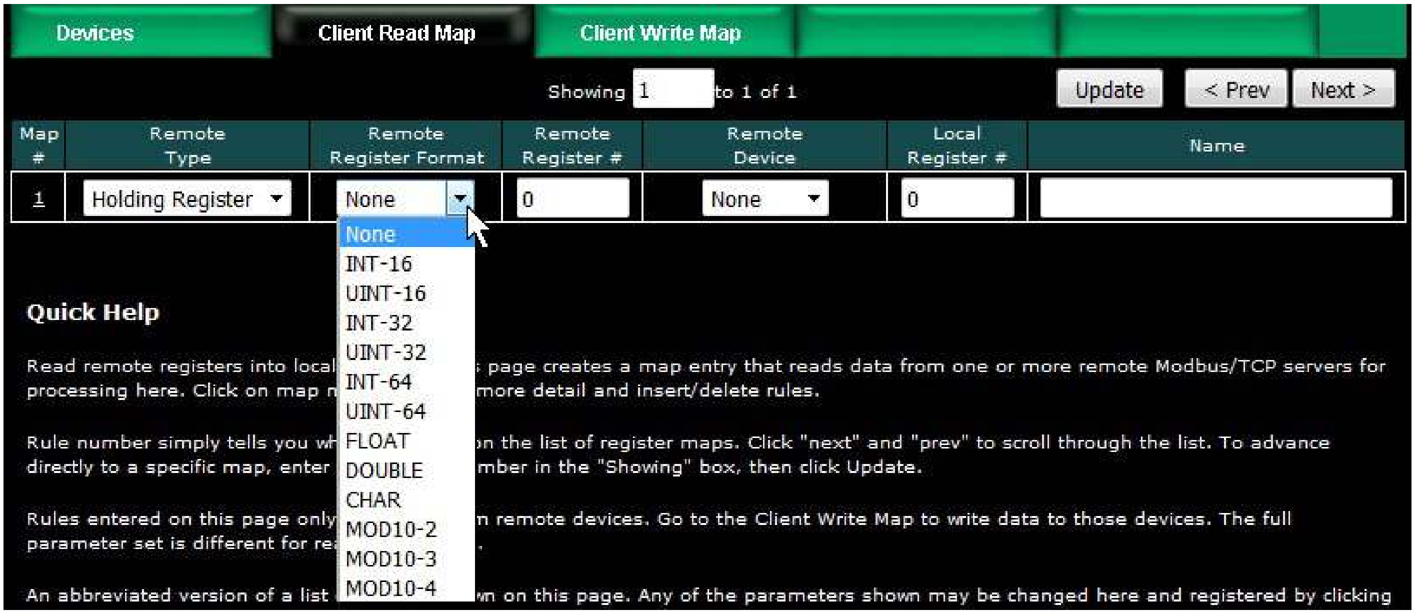
Showing 1 to 1 of 1

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

Quic

Read remote registers into local registers. This page creates a map entry that reads data from one or more remote Modbus/TCP servers for

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



Showing 1 to 1 of 1

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

Quick Help

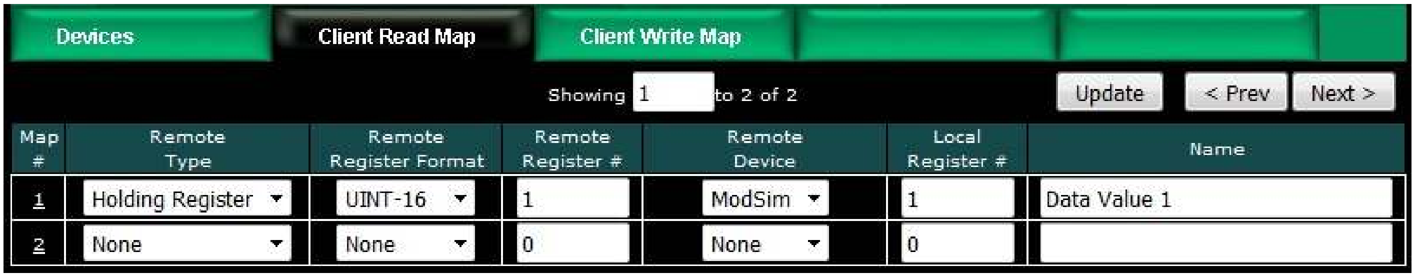
Read remote registers into local ; page creates a map entry that reads data from one or more remote Modbus/TCP servers for processing here. Click on map n more detail and insert/delete rules.

Rule number simply tells you wh on the list of register maps. Click "next" and "prev" to scroll through the list. To advance directly to a specific map, enter mber in the "Showing" box, then click Update.

Rules entered on this page only n remote devices. Go to the Client Write Map to write data to those devices. The full parameter set is different for re .

An abbreviated version of a list wn on this page. Any of the parameters shown may be changed here and registered by clicking

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.)



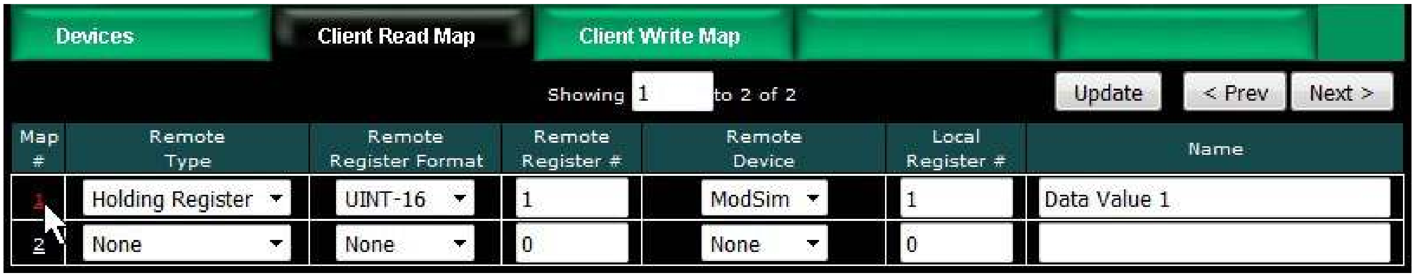
Showing 1 to 2 of 2

Map #	Remote Type	Remote Register Format	Remote Register #	Remote Device	Local Register #	Name
1	Holding Register	UINT-16	1	ModSim	1	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.

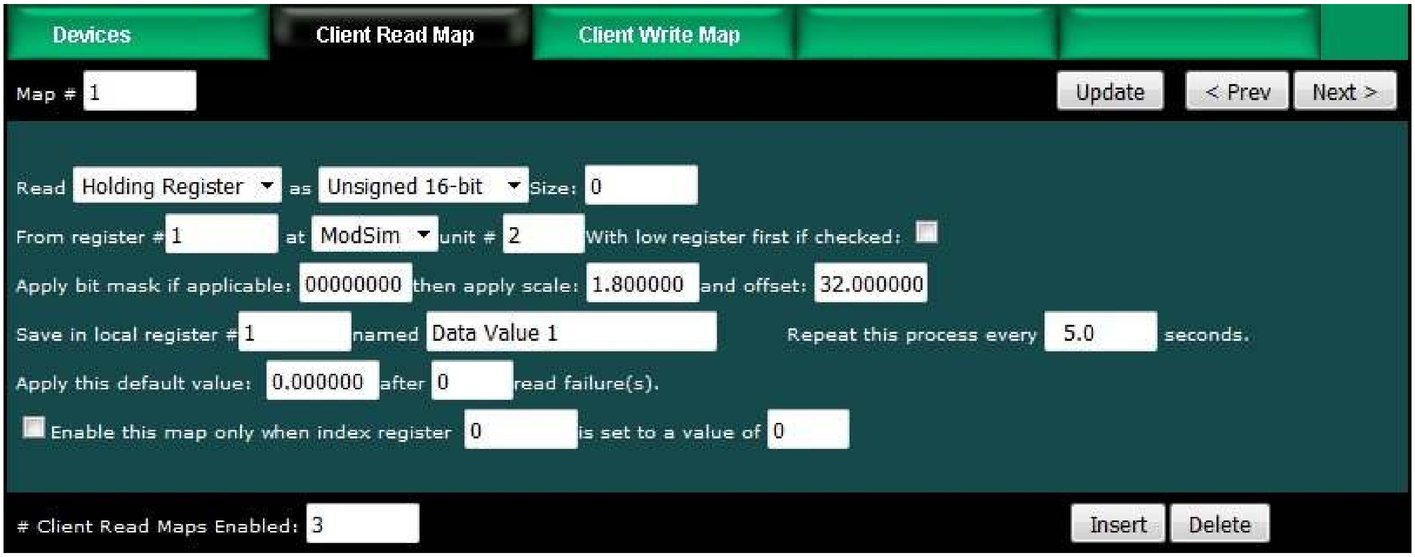
The Local Register is where data read from the remote Modbus TCP server will be stored locally in the Babel Buster IoT Gateway. If the local register data format does not match what you are reading from the Modbus device, the data will be converted automatically when it is read.



Showing 1 to 2 of 2

Map #	Remote Type	Remote Register Format	Remote Register #	Remote Device	Local Register #	Name
1	Holding Register	UINT-16	1	ModSim	1	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.



Map # 1

Read Holding Register as Unsigned 16-bit Size: 0

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

Apply bit mask if applicable: 00000000 then apply scale: 1.800000 and offset: 32.000000

Save in local register # 1 named Data Value 1 Repeat this process every 5.0 seconds.

Apply this default value: 0.000000 after 0 read failure(s).

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

Client Read Maps Enabled: 3

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, Babel Buster 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.

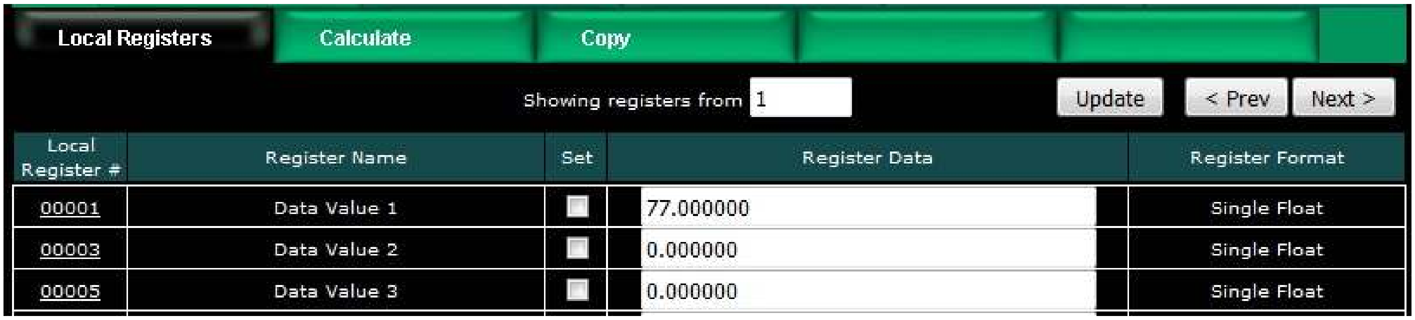
You have the option of making this Read Map conditional. If an index register number is provided and the Enable box is checked, then this read map will only be executed when the index register (local register) contains the value given. This allows multiple read maps to supply data to the same local register 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.



Showing registers from 1

Local Register #	Register Name	Set	Register Data	Register Format
00001	Data Value 1	□	77.000000	Single Float
00003	Data Value 2	□	0.000000	Single Float
00005	Data Value 3	□	0.000000	Single Float

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).

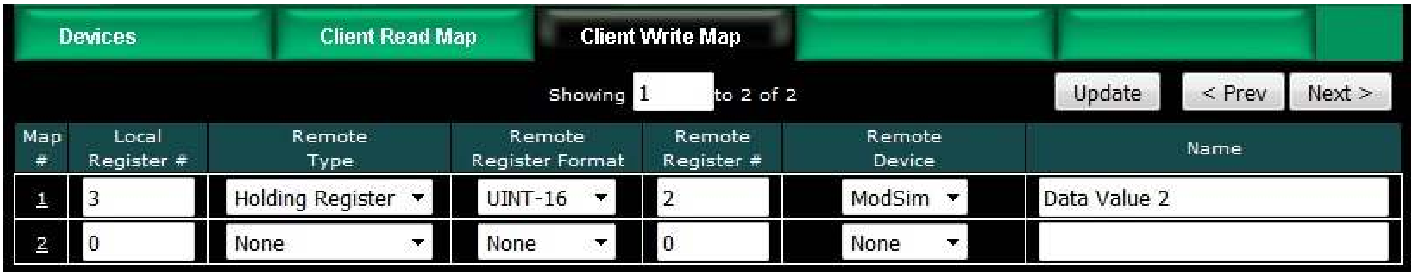
It is common for Modbus devices to pack a number of status bits into a single holding register. In order to do meaningful things based on a single bit, it is sometimes necessary to split that register into multiple local registers. Babel Buster supports this requirement by providing an optional bit mask.

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 Babel Buster 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

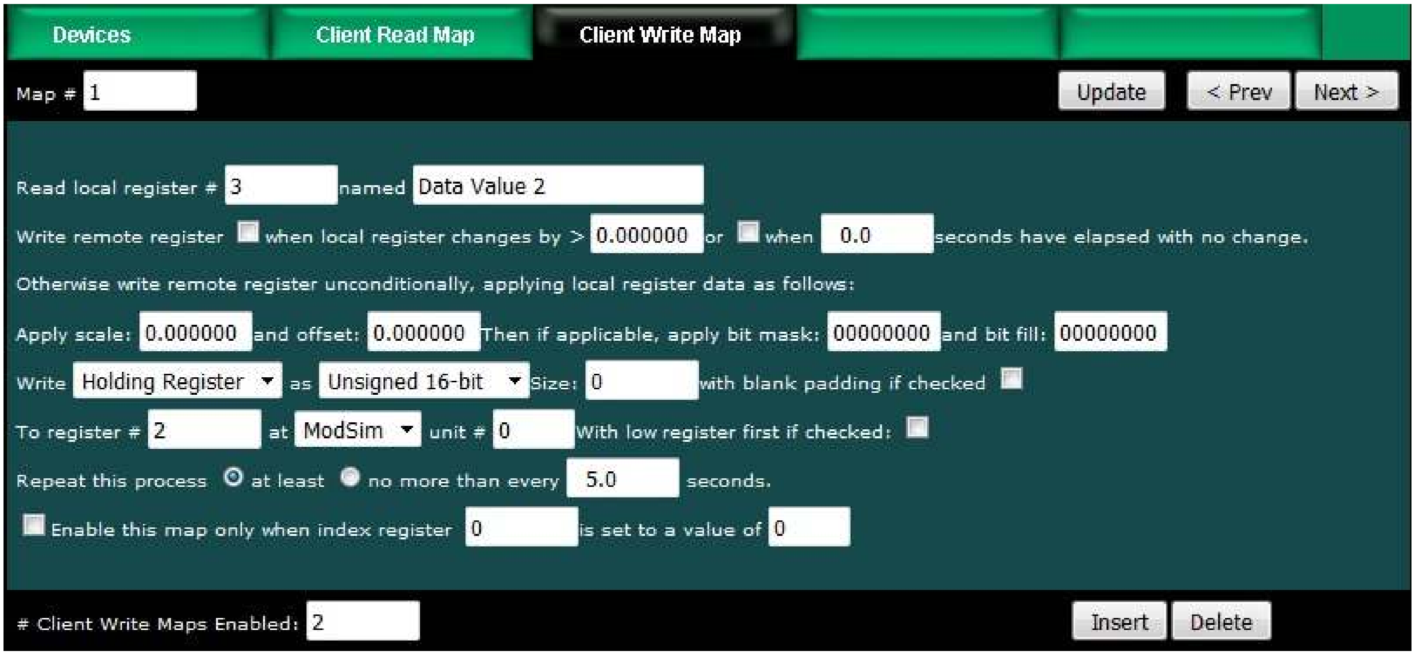
Getting the gateway to write registers to another Modbus device requires setting up a "Write Map" as shown here. Much of the Write Map is configured the same as a Read Map.



Showing 1 to 2 of 2

Map #	Local Register #	Remote Type	Remote Register Format	Remote Register #	Remote Device	Name
1	3	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

Read local register # 3 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: 00000000 and bit fill: 00000000

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

To register # 2 at ModSim unit # 0 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

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.

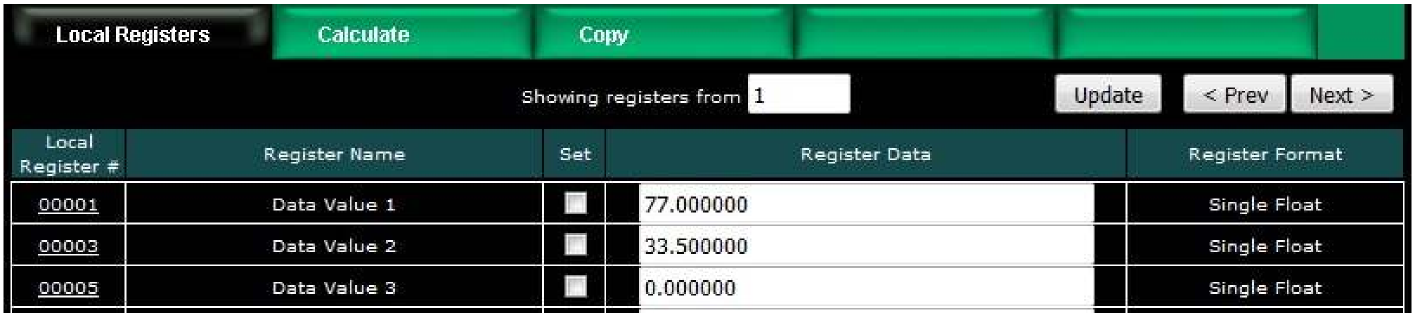
Modbus protocol treats all holding registers as strictly 16-bit registers. To accommodate 32-bit or longer data, Modbus devices use multiple consecutive "registers" to hold the data. There is no standardization of whether the least significant part of the data comes first or last. Therefore, Babel Buster 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.

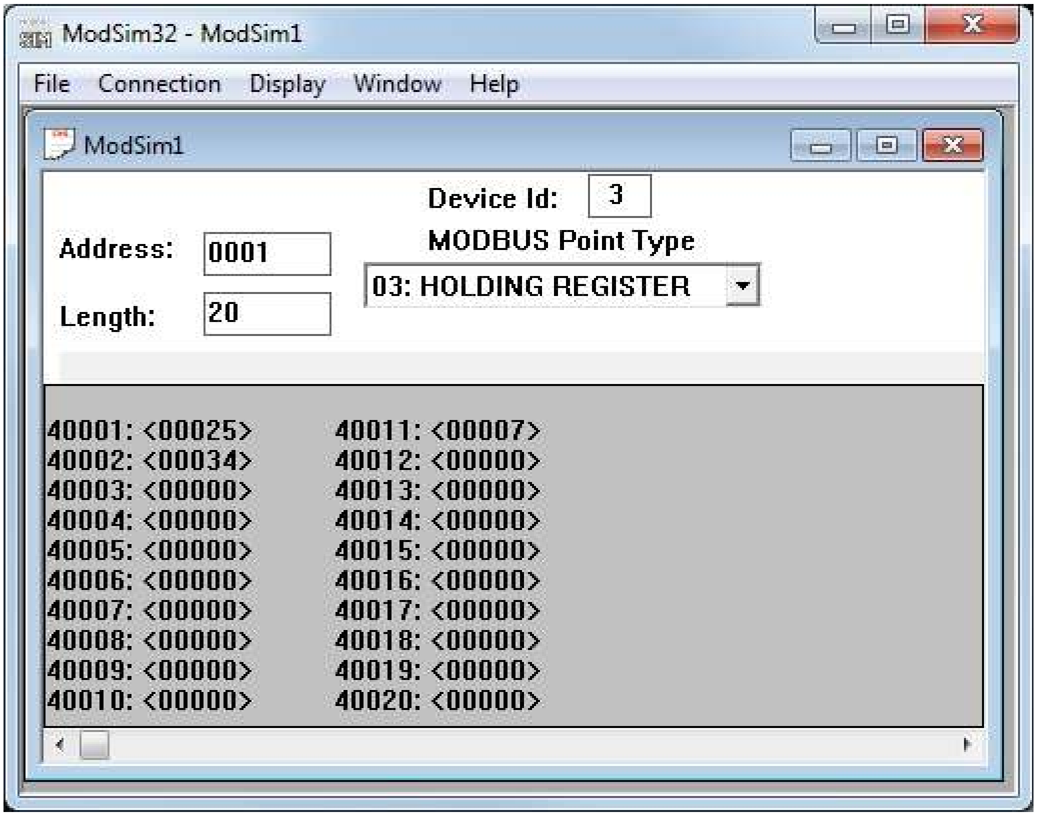
Selecting "none" for remote type effectively deletes the map even though it will still appear in the list until deleted. Unused maps at the end of the list will always show none as the type. If you wish to prevent these from being displayed, reduce the number of maps enabled.



Showing registers from 1

Local Register #	Register Name	Set	Register Data	Register Format
00001	Data Value 1	□	77.000000	Single Float
00003	Data Value 2	□	33.500000	Single Float
00005	Data Value 3	□	0.000000	Single Float

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.



Device Id: 3

Address: 0001

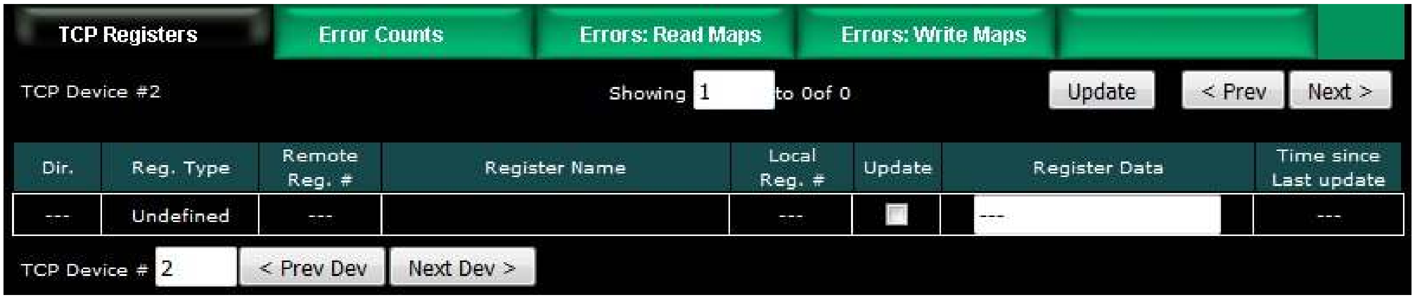
Length: 20

MODBUS Point Type: 03: HOLDING REGISTER

```
40001: <00025>      40011: <00007>
40002: <00034>      40012: <00000>
40003: <00000>      40013: <00000>
40004: <00000>      40014: <00000>
40005: <00000>      40015: <00000>
40006: <00000>      40016: <00000>
40007: <00000>      40017: <00000>
40008: <00000>      40018: <00000>
40009: <00000>      40019: <00000>
40010: <00000>      40020: <00000>
```

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.



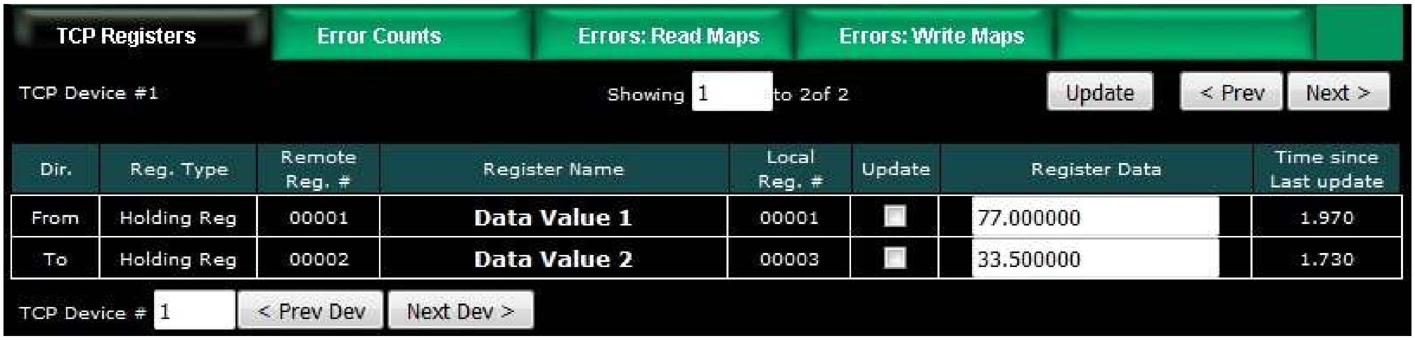
TCP Device #2

Showing 1 to 0of 0

Dir.	Reg. Type	Remote Reg. #	Register Name	Local Reg. #	Update	Register Data	Time since Last update
---	Undefined	---		---	□	---	---

TCP Device # 2

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.



TCP Device #1

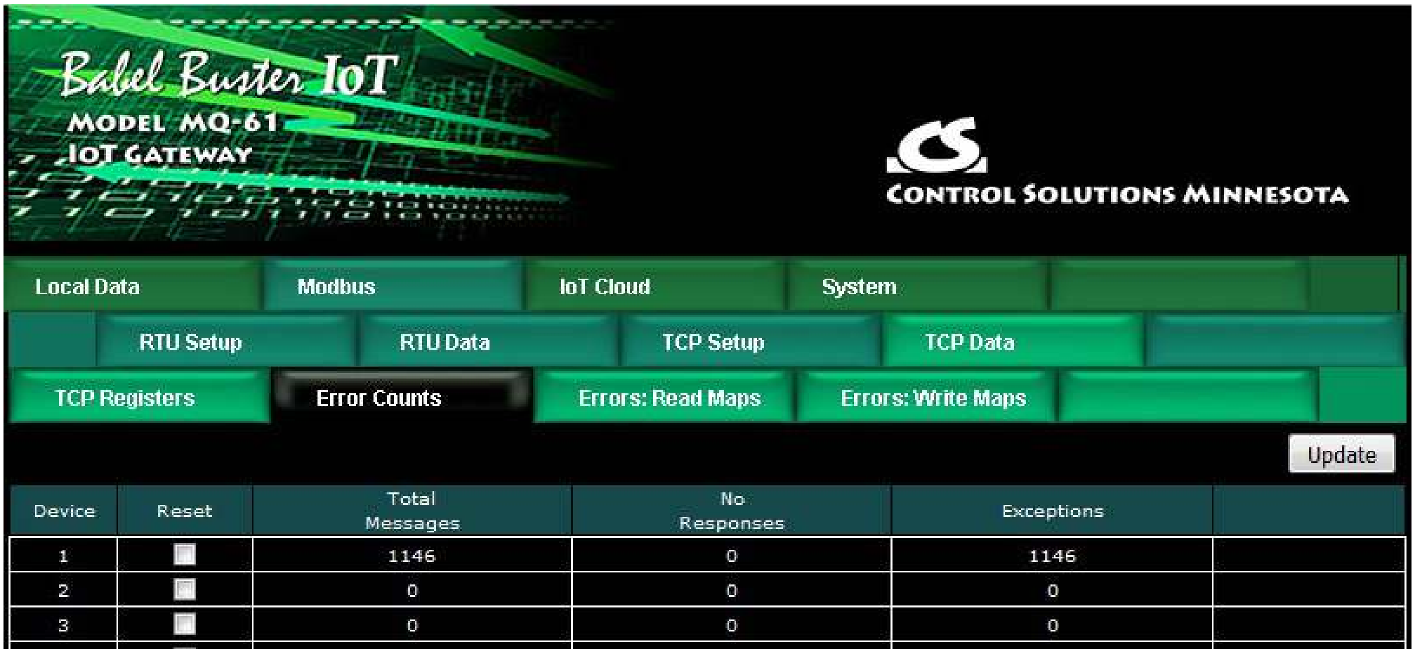
Showing 1 to 2of 2

Dir.	Reg. Type	Remote Reg. #	Register Name	Local Reg. #	Update	Register Data	Time since Last update
From	Holding Reg	00001	**Data Value 1**	00001	□	77.000000	1.970
To	Holding Reg	00002	**Data Value 2**	00003	□	33.500000	1.730

TCP Device # 1

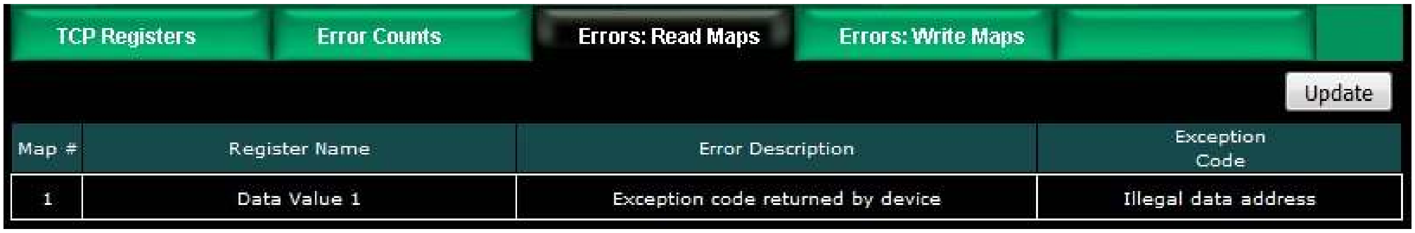
6.5 Modbus TCP Errors

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.



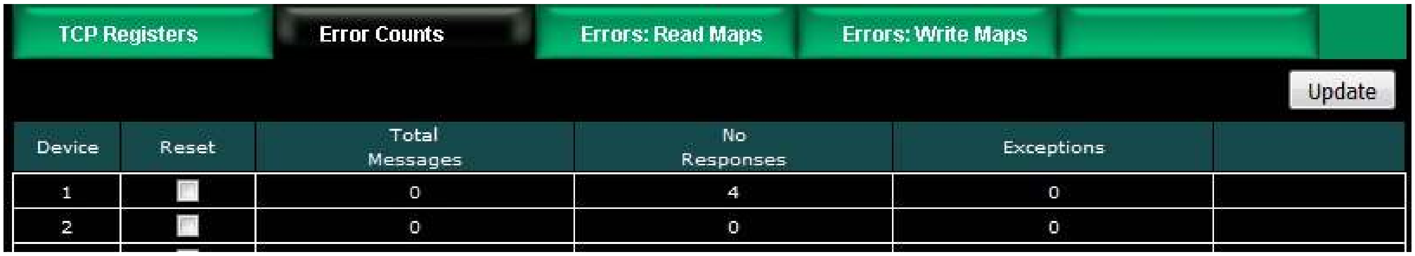
Device	Reset	Total Messages	No Responses	Exceptions	
1	□	1146	0	1146	
2	□	0	0	0	
3	□	0	0	0	

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.



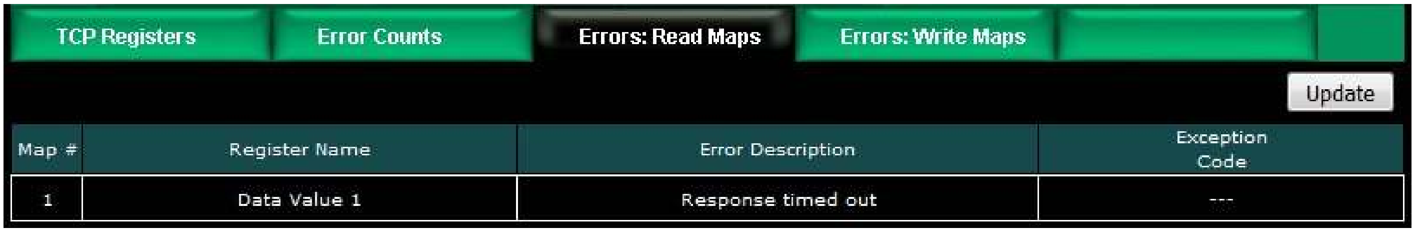
Map #	Register Name	Error Description	Exception Code
1	Data Value 1	Exception code returned by device	Illegal data address

If you see total messages of zero and a "no responses" count greater than zero, it means the Babel Buster 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.



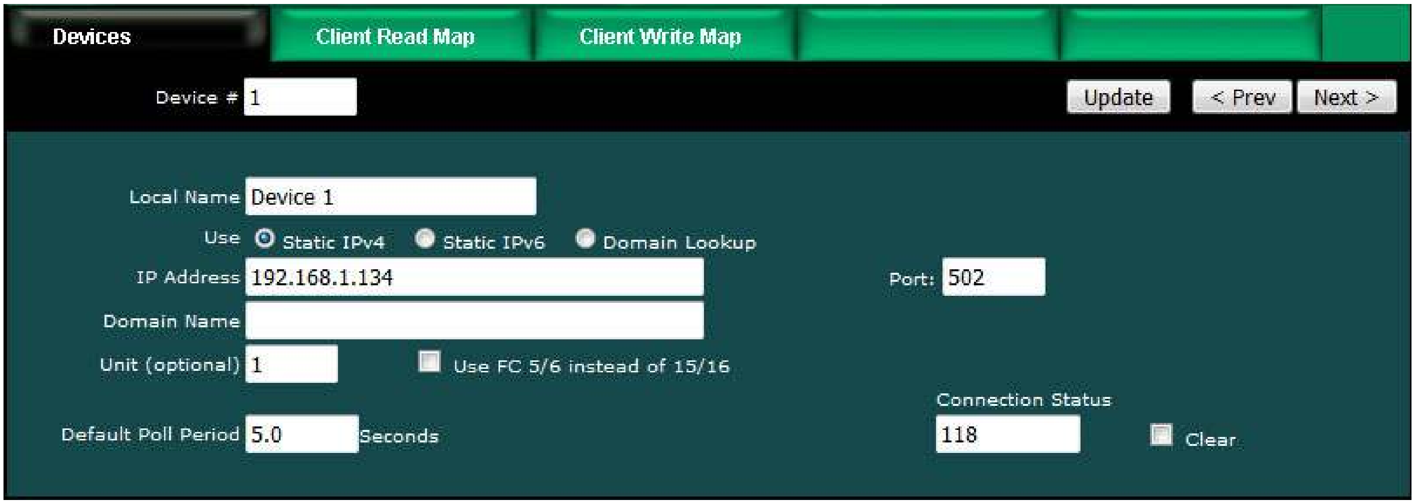
Device	Reset	Total Messages	No Responses	Exceptions
1	□	0	4	0
2	□	0	0	0

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.



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.



Device # 1

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 118 □ Clear

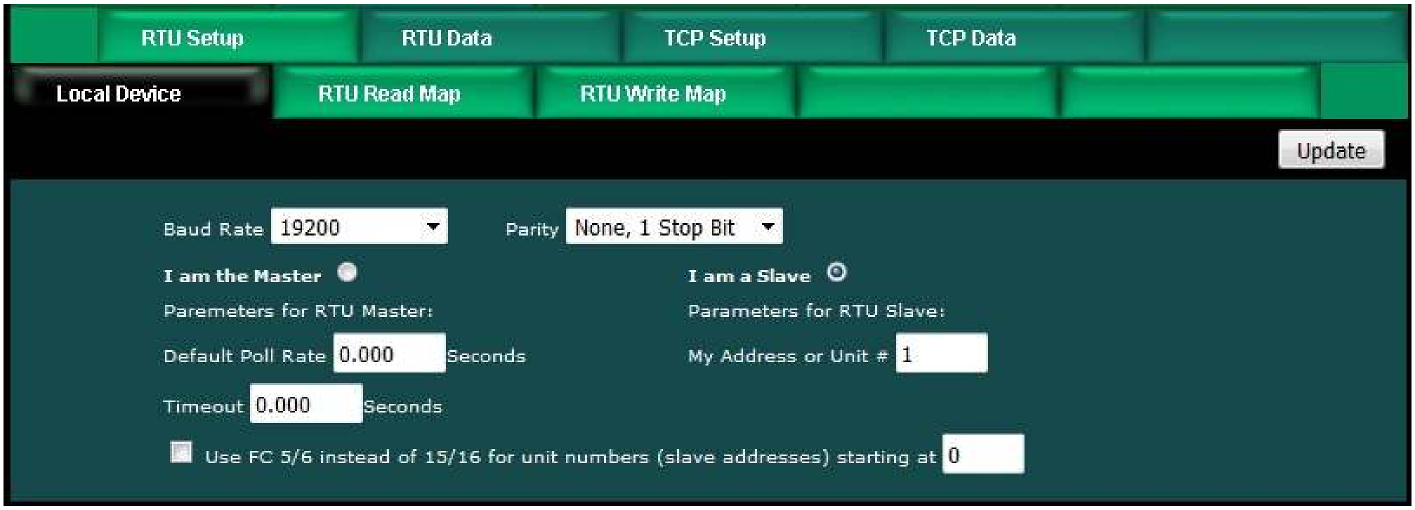
Connection status codes you may see include:

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 Gateway 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.



Baud Rate 19200 Parity None, 1 Stop Bit

I am the Master ○ I am a Slave ●

Paremeters for RTU Master:

Default Poll Rate 0.000 Seconds

Timeout 0.000 Seconds

Parameters for RTU Slave:

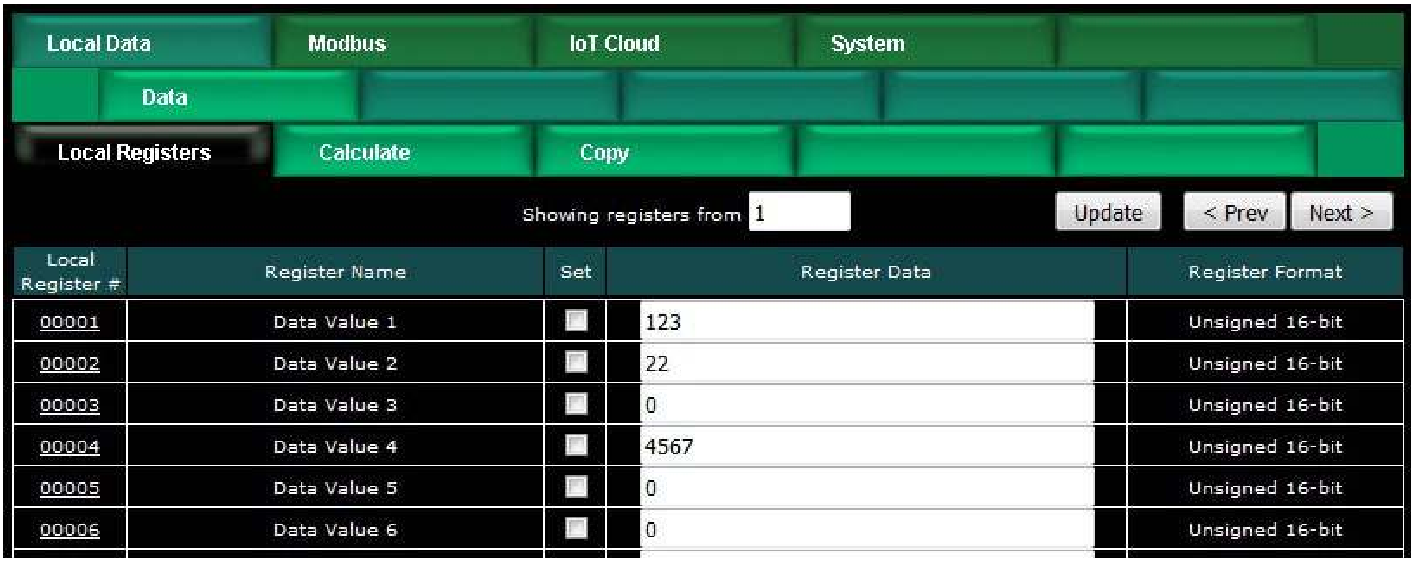
My Address or Unit # 1

□ Use FC 5/6 instead of 15/16 for unit numbers (slave addresses) starting at 0

7.2 Modbus RTU Slave Register Map

The local registers in the Babel Buster IoT Gateway 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.

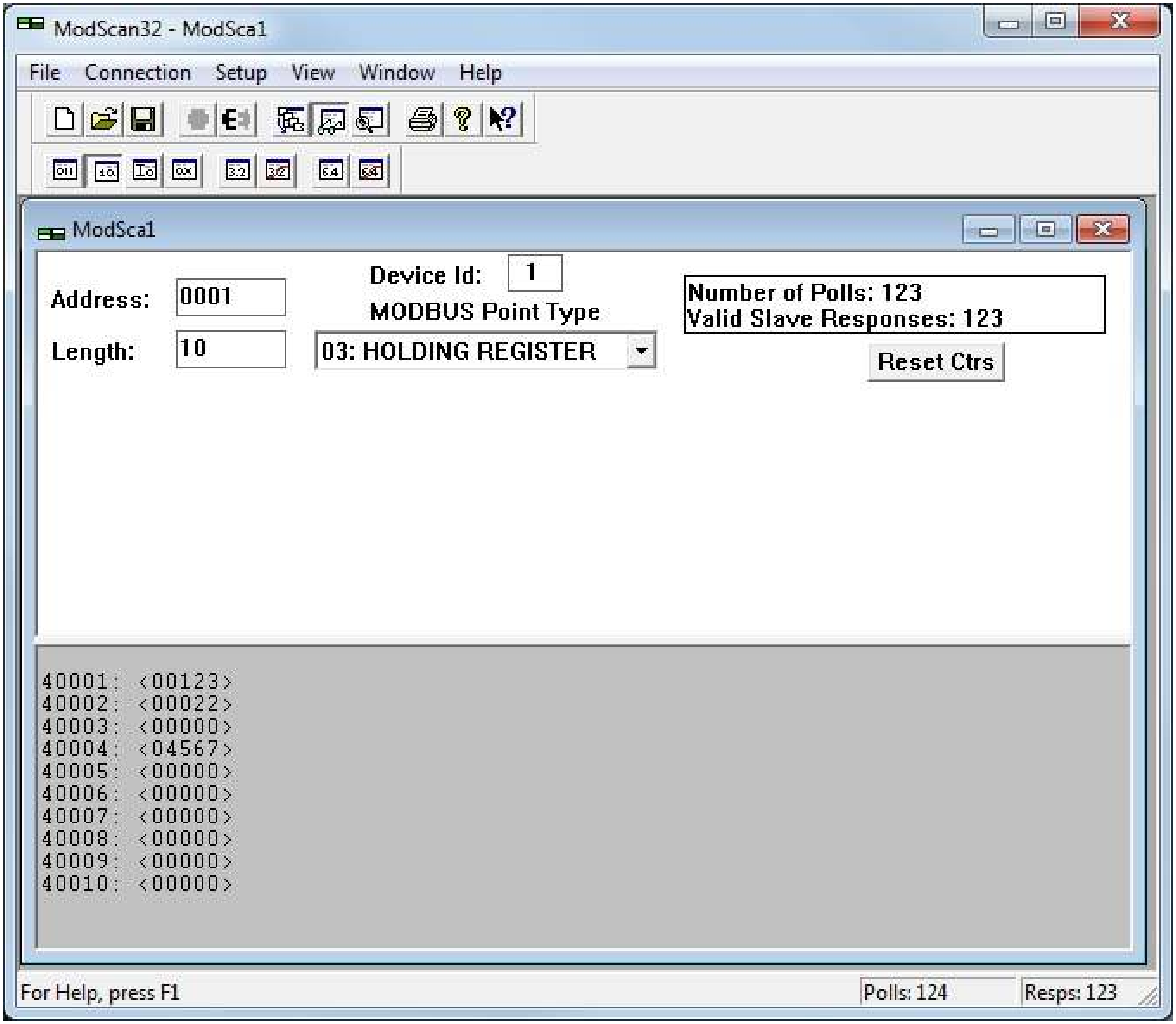
The register numbers that the remote Modbus RTU master should read or write are simply those shown in the first column on the Local Registers page.



Showing registers from 1

Local Register #	Register Name	Set	Register Data	Register Format
00001	Data Value 1	□	123	Unsigned 16-bit
00002	Data Value 2	□	22	Unsigned 16-bit
00003	Data Value 3	□	0	Unsigned 16-bit
00004	Data Value 4	□	4567	Unsigned 16-bit
00005	Data Value 5	□	0	Unsigned 16-bit
00006	Data Value 6	□	0	Unsigned 16-bit

If the local registers are defined as unsigned 16-bit and contain the data illustrated above, then ModScan will read them as illustrated below.



Address: 0001

Length: 10

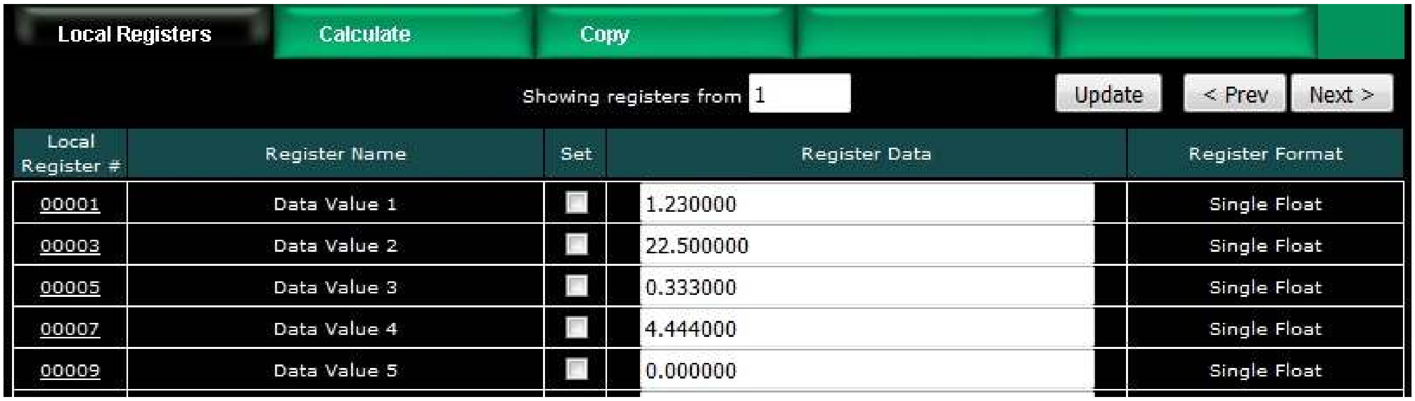
Device Id: 1

MODBUS Point Type: 03: HOLDING REGISTER

Number of Polls: 123
Valid Slave Responses: 123

```
40001: <00123>
40002: <00022>
40003: <00000>
40004: <04567>
40005: <00000>
40006: <00000>
40007: <00000>
40008: <00000>
40009: <00000>
40010: <00000>
```

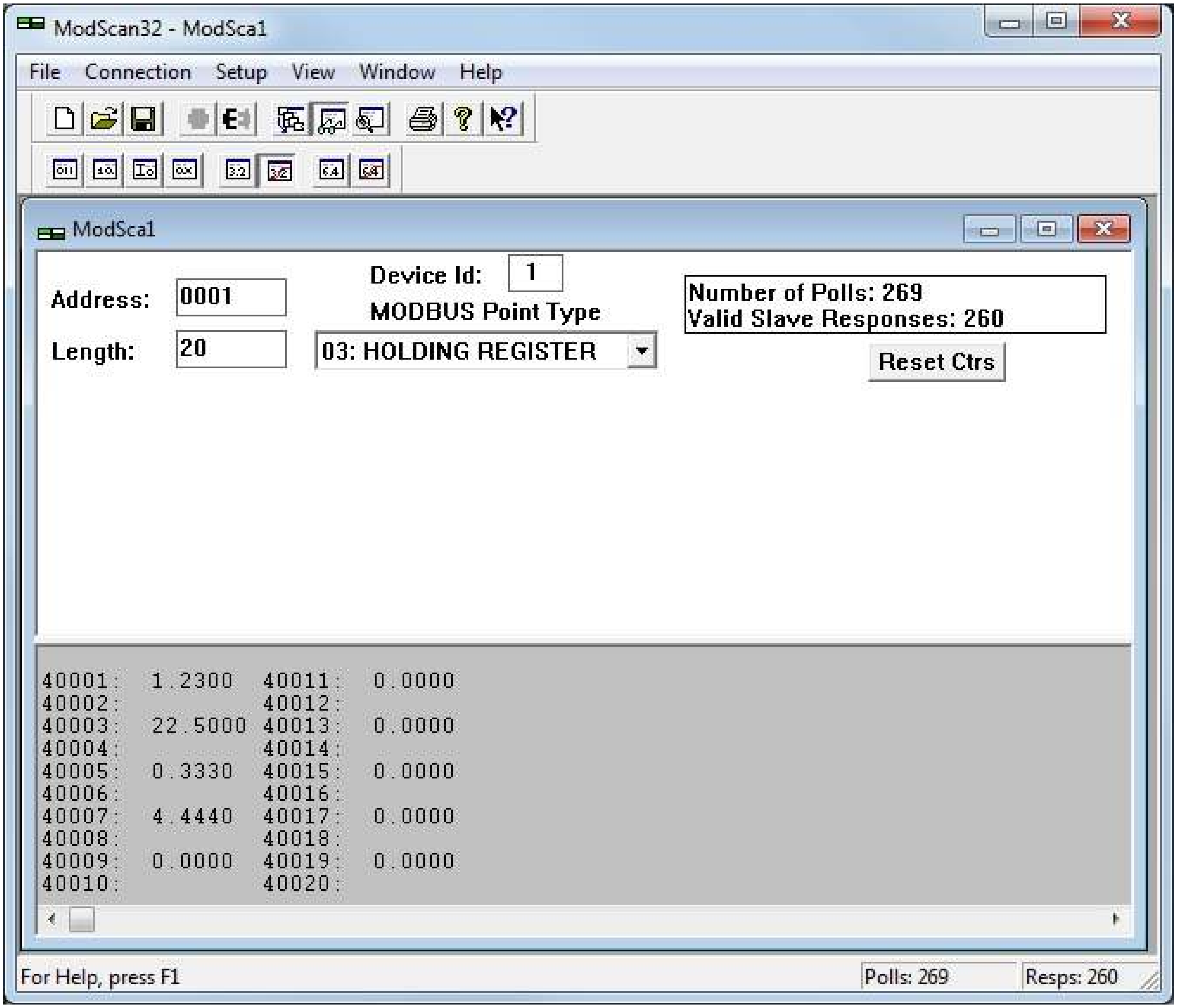
Polls: 124 Resps: 123



Showing registers from 1

Local Register #	Register Name	Set	Register Data	Register Format
00001	Data Value 1	☐	1.230000	Single Float
00003	Data Value 2	☐	22.500000	Single Float
00005	Data Value 3	☐	0.333000	Single Float
00007	Data Value 4	☐	4.444000	Single Float
00009	Data Value 5	☐	0.000000	Single Float

If the local registers are defined as floating point and contain the data illustrated above, then ModScan will read them as illustrated below. Note that registers defined as having data formats greater than 16 bits must be read on the proper boundaries. You are not allowed to read half of a floating point number or just part of any other larger data value. If you were to attempt to read register #2 in the this example, you would get an exception error. This also means you cannot read just 2 characters in the middle of a larger character string. When the register is defined as character string, you must access the entire character string as a collective set of registers (i.e. read/write multiple holding registers in a single request).



Address: 0001

Device Id: 1

Length: 20

MODBUS Point Type: 03: HOLDING REGISTER

Number of Polls: 269
Valid Slave Responses: 260

```
40001:  1.2300  40011:  0.0000
40002:          40012:
40003:  22.5000 40013:  0.0000
40004:          40014:
40005:  0.3330  40015:  0.0000
40006:          40016:
40007:  4.4440  40017:  0.0000
40008:          40018:
40009:  0.0000  40019:  0.0000
40010:          40020:
```

For Help, press F1 Polls: 269 Resps: 260

7.3 Modbus RTU Slave Diagnostic

The Babel Buster IoT Gateway brand new out of the box will have no registers configured, and will be configured as Modbus RTU slave, 9600 baud, slave address 1. Although no registers 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".

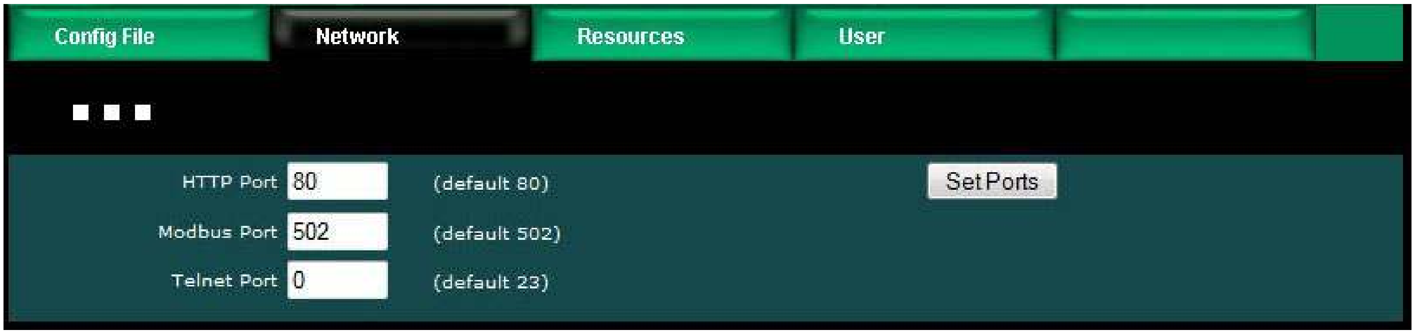
8. Configuring Gateway as a Modbus TCP Server

8.1 Modbus TCP Device Configuration

There is really little to do to configure the Babel Buster IoT Gateway 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.



HTTP Port 80 (default 80)

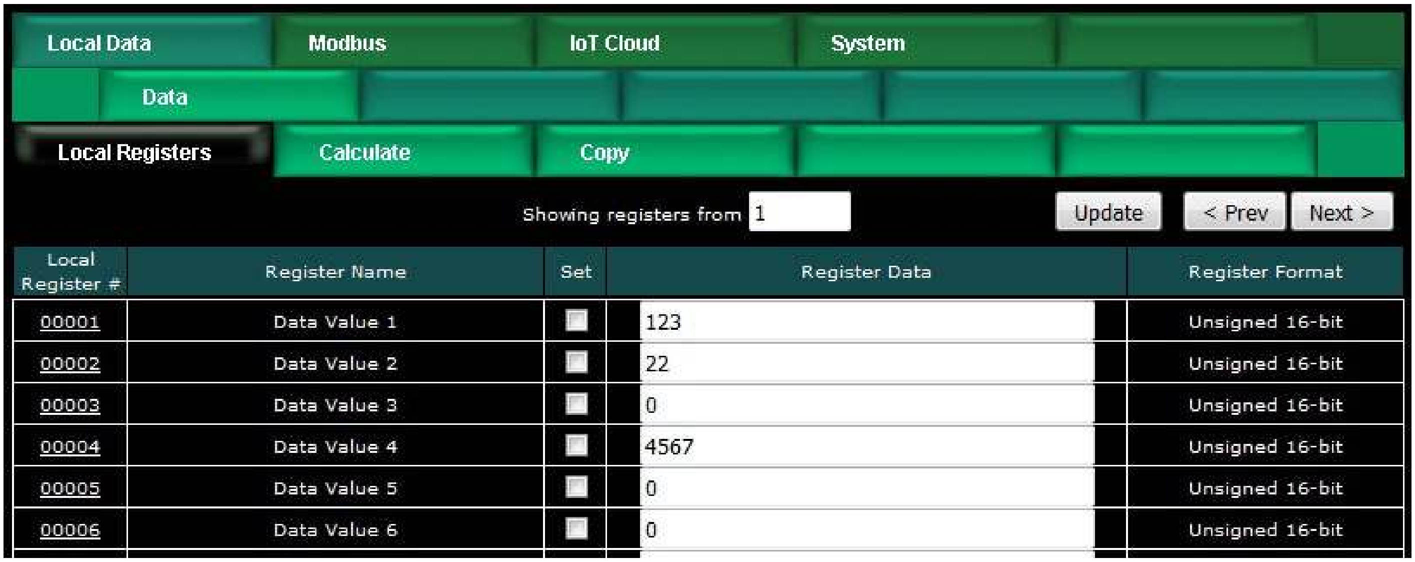
Modbus Port 502 (default 502)

Telnet Port 0 (default 23)

8.2 Modbus TCP Server Register Map

The local registers in the Babel Buster IoT Gateway 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.

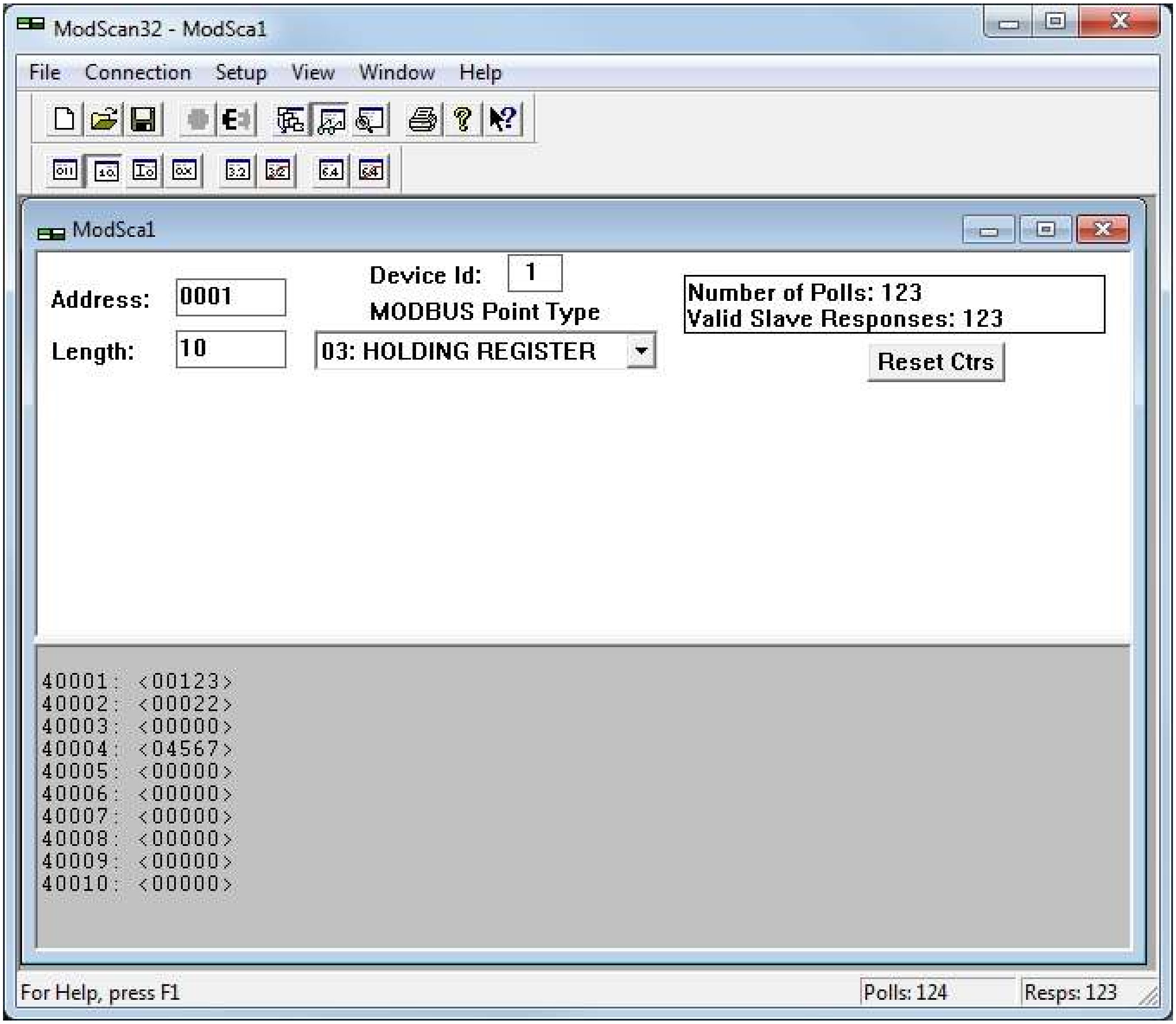
The register numbers that the remote Modbus TCP client should read or write are simply those shown in the first column on the Local Registers page.



Showing registers from 1

Local Register #	Register Name	Set	Register Data	Register Format
00001	Data Value 1	□	123	Unsigned 16-bit
00002	Data Value 2	□	22	Unsigned 16-bit
00003	Data Value 3	□	0	Unsigned 16-bit
00004	Data Value 4	□	4567	Unsigned 16-bit
00005	Data Value 5	□	0	Unsigned 16-bit
00006	Data Value 6	□	0	Unsigned 16-bit

If the local registers are defined as unsigned 16-bit and contain the data illustrated above, then ModScan will read them as illustrated below.



Address: 0001

Device Id: 1

MODBUS Point Type

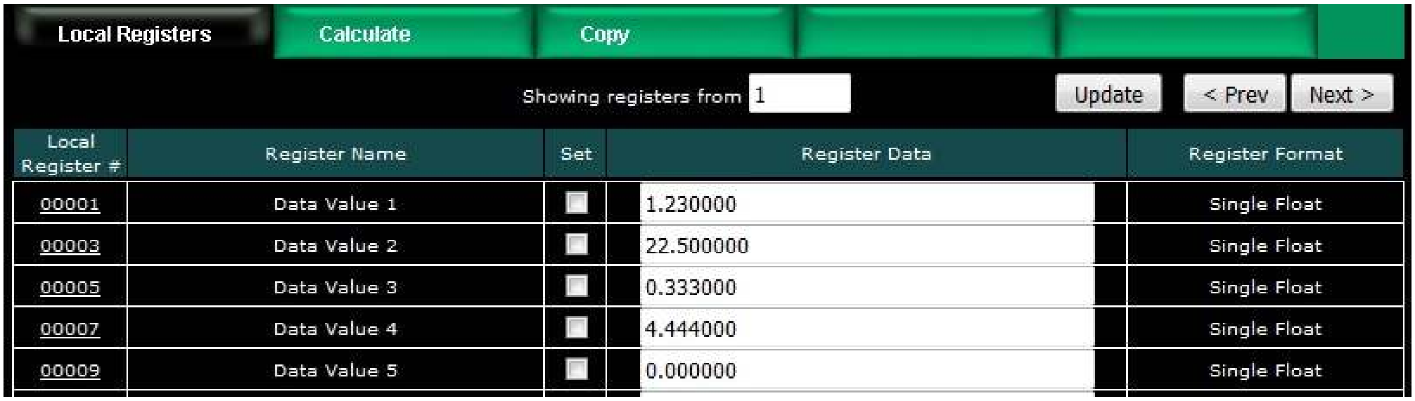
Length: 10

03: HOLDING REGISTER

Number of Polls: 123
Valid Slave Responses: 123

```
40001: <00123>
40002: <00022>
40003: <00000>
40004: <04567>
40005: <00000>
40006: <00000>
40007: <00000>
40008: <00000>
40009: <00000>
40010: <00000>
```

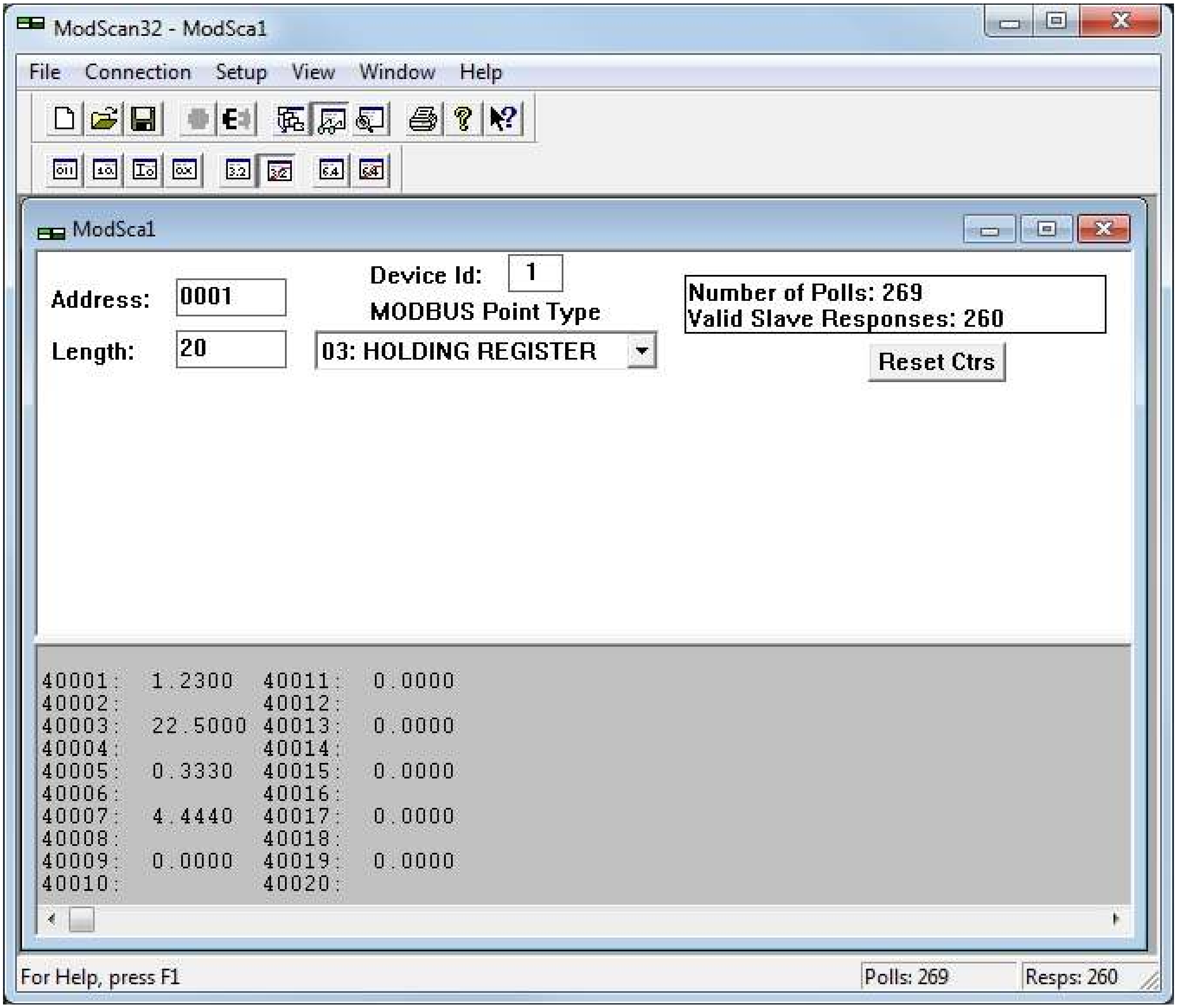
For Help, press F1 Polls: 124 Resps: 123



Showing registers from 1

Local Register #	Register Name	Set	Register Data	Register Format
00001	Data Value 1	□	1.230000	Single Float
00003	Data Value 2	□	22.500000	Single Float
00005	Data Value 3	□	0.333000	Single Float
00007	Data Value 4	□	4.444000	Single Float
00009	Data Value 5	□	0.000000	Single Float

If the local registers are defined as floating point and contain the data illustrated above, then ModScan will read them as illustrated below. Note that registers defined as having data formats greater than 16 bits must be read on the proper boundaries. You are not allowed to read half of a floating point number or just part of any other larger data value. If you were to attempt to read register #2 in the this example, you would get an exception error. This also means you cannot read just 2 characters in the middle of a larger character string. When the register is defined as character string, you must access the entire character string as a collective set of registers (i.e. read/write multiple holding registers in a single request).



Address: 0001
Length: 20
Device Id: 1
MODBUS Point Type
03: HOLDING REGISTER
Number of Polls: 269
Valid Slave Responses: 260

```
40001:  1.2300  40011:  0.0000
40002:          40012:
40003:  22.5000 40013:  0.0000
40004:          40014:
40005:  0.3330  40015:  0.0000
40006:          40016:
40007:  4.4440  40017:  0.0000
40008:          40018:
40009:  0.0000  40019:  0.0000
40010:          40020:
```

Polls: 269 Resps: 260

8.3 Modbus TCP Server Diagnostic

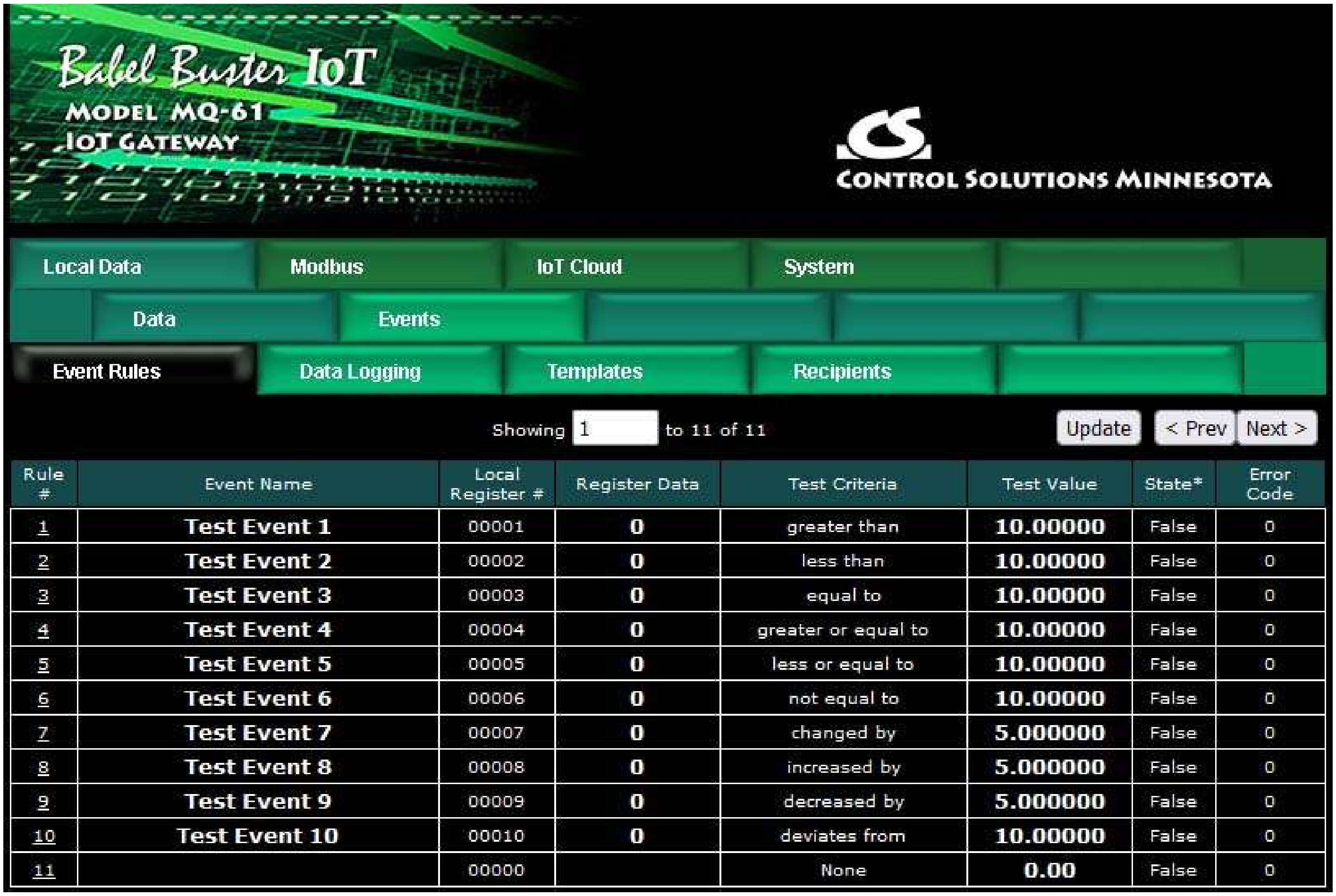
The Babel Buster brand new out of the box will have no registers configured. Although no registers are configured, there will be a single holding register accessible for diagnostic purposes, at register number 8801 (or 48801 if using Modicon notation). The content of this register will be firmware revision 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".

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.



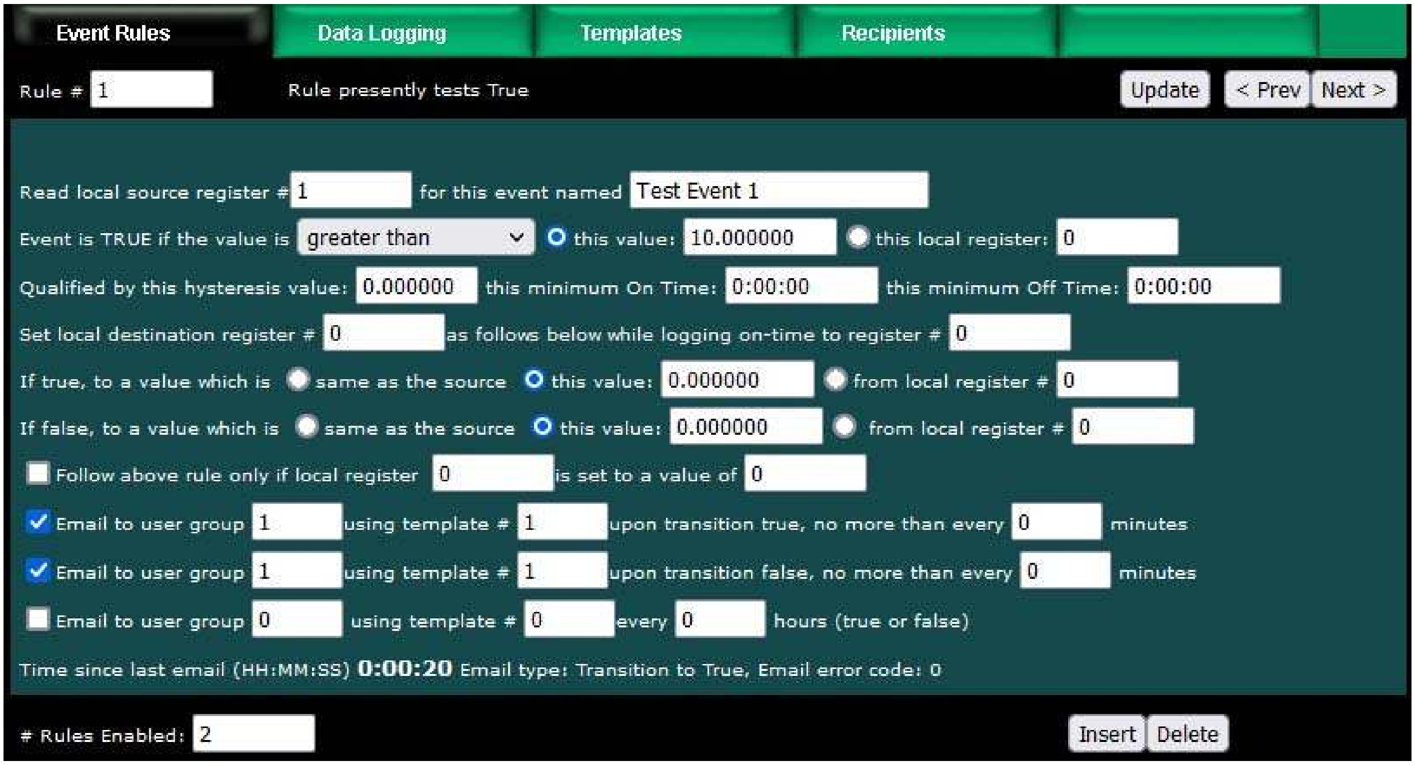
Showing 1 to 11 of 11

Rule #	Event Name	Local Register #	Register Data	Test Criteria	Test Value	State*	Error Code
1	Test Event 1	00001	0	greater than	10.00000	False	0
2	Test Event 2	00002	0	less than	10.00000	False	0
3	Test Event 3	00003	0	equal to	10.00000	False	0
4	Test Event 4	00004	0	greater or equal to	10.00000	False	0
5	Test Event 5	00005	0	less or equal to	10.00000	False	0
6	Test Event 6	00006	0	not equal to	10.00000	False	0
7	Test Event 7	00007	0	changed by	5.000000	False	0
8	Test Event 8	00008	0	increased by	5.000000	False	0
9	Test Event 9	00009	0	decreased by	5.000000	False	0
10	Test Event 10	00010	0	deviates from	10.00000	False	0
11		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.



Rule # 1 Rule presently tests True

Read local source register # 1 for this event named Test Event 1

Event is TRUE if the value is greater than ● this value: 10.000000 ○ 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

Set local destination register # 0 as follows below while logging on-time to register # 0

If true, to a value which is ○ same as the source ● this value: 0.000000 ○ from local register # 0

If false, to a value which is ○ same as the source ● this value: 0.000000 ○ from local register # 0

☐ Follow above rule only if local register 0 is set to a value of 0

☑ Email to user group 1 using template # 1 upon transition true, no more than every 0 minutes

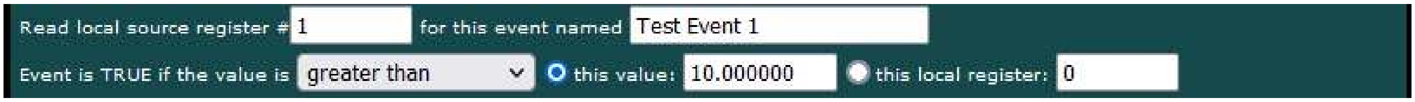
☑ Email to user group 1 using template # 1 upon transition false, no more than every 0 minutes

☐ Email to user group 0 using template # 0 every 0 hours (true or false)

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

Rules Enabled: 2

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.



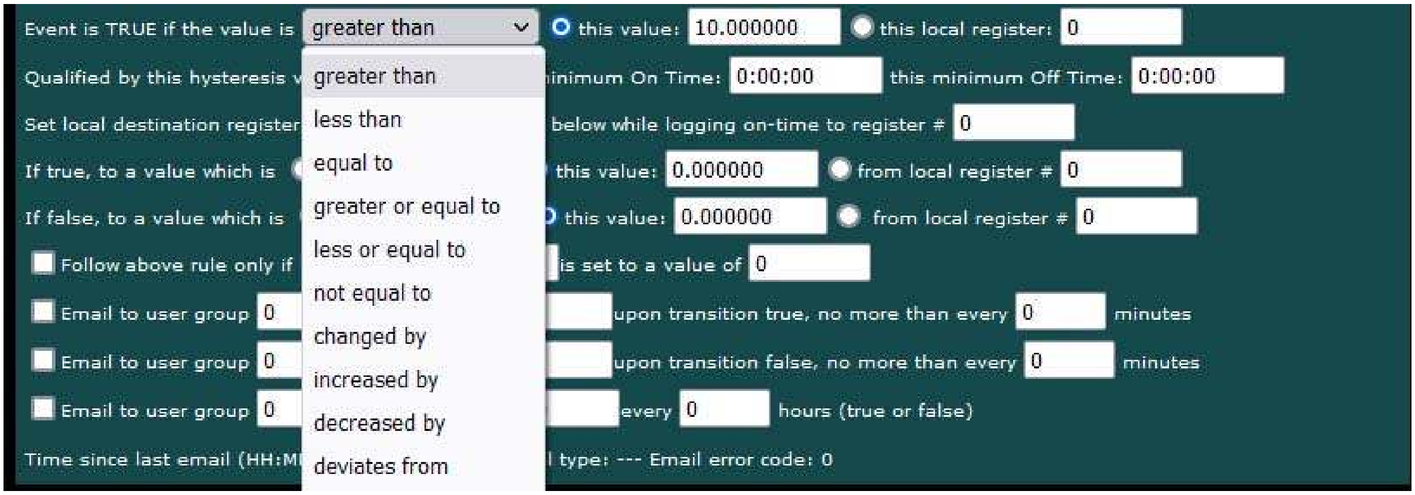
Read local source register # 1 for this event named Test Event 1

Event is TRUE if the value is greater than ● this value: 10.000000 ○ 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.


Event is TRUE if the value is greater than ● this value: 10.000000 ○ this local register: 0

Qualified by this hysteresis v inimum On Time: 0:00:00 this minimum Off Time: 0:00:00

Set local destination register below while logging on-time to register # 0

If true, to a value which is ● this value: 0.000000 ○ from local register # 0

If false, to a value which is ● this value: 0.000000 ○ from local register # 0

□ Follow above rule only if is set to a value of 0

□ Email to user group 0 upon transition true, no more than every 0 minutes

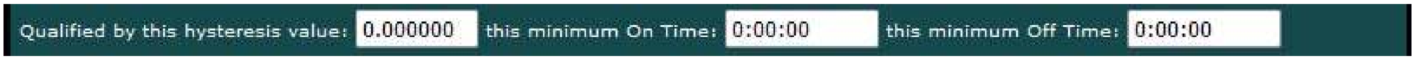
□ Email to user group 0 upon transition false, no more than every 0 minutes

□ Email to user group 0 every 0 hours (true or false)

Time since last email (HH:M l type: --- Email error code: 0

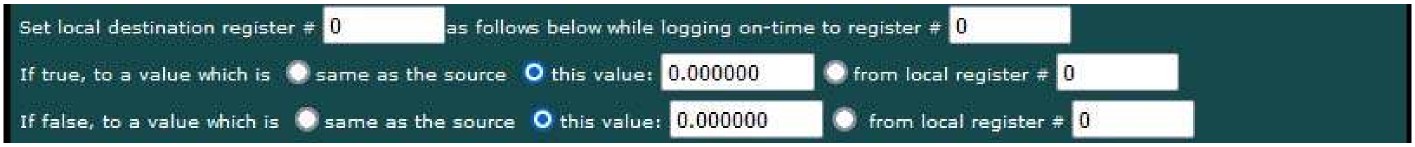
The possible test types are illustrated above. Some tests need further explanation. The "Changed by" test means amount of change since the last event transition to true. If the local register has changed by the value specified as "this value" or the value contained in the local register referenced, the 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.


Qualified by this hysteresis value: 0.000000 this minimum On Time: 0:00:00 this minimum Off Time: 0:00:00

Quaifications 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.


Set local destination register # 0 as follows below while logging on-time to register # 0

If true, to a value which is ○ same as the source ● this value: 0.000000 ○ from local register # 0

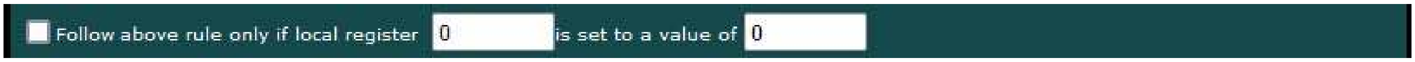
If false, to a value which is ○ same as the source ● this value: 0.000000 ○ from local register # 0

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.

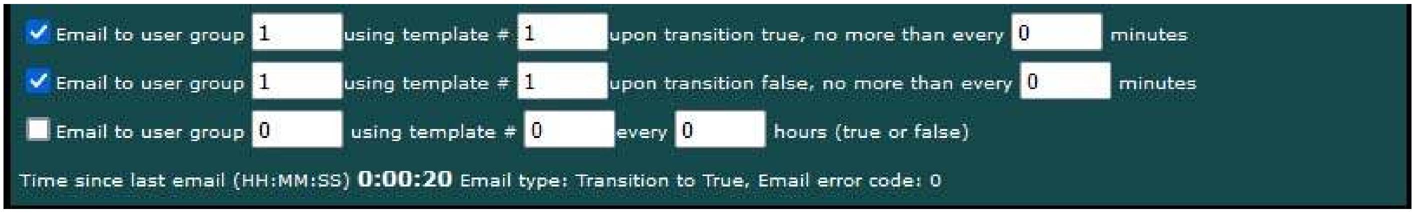
The first line after the destination register number is the response that will be taken when the condition is true, and the following line is the response that will be taken when the condition is false. Either the source register is copied, a fixed value is applied, or another register is used to provide 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 0 is set to a value of 0

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 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

□ Email to user group 0 using template # 0 every 0 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.

Babel Buster IoT
MODEL MQ-61
MODEL BB3-6101-MQ
IOT GATEWAY

CONTROL SOLUTIONS MINNESOTA

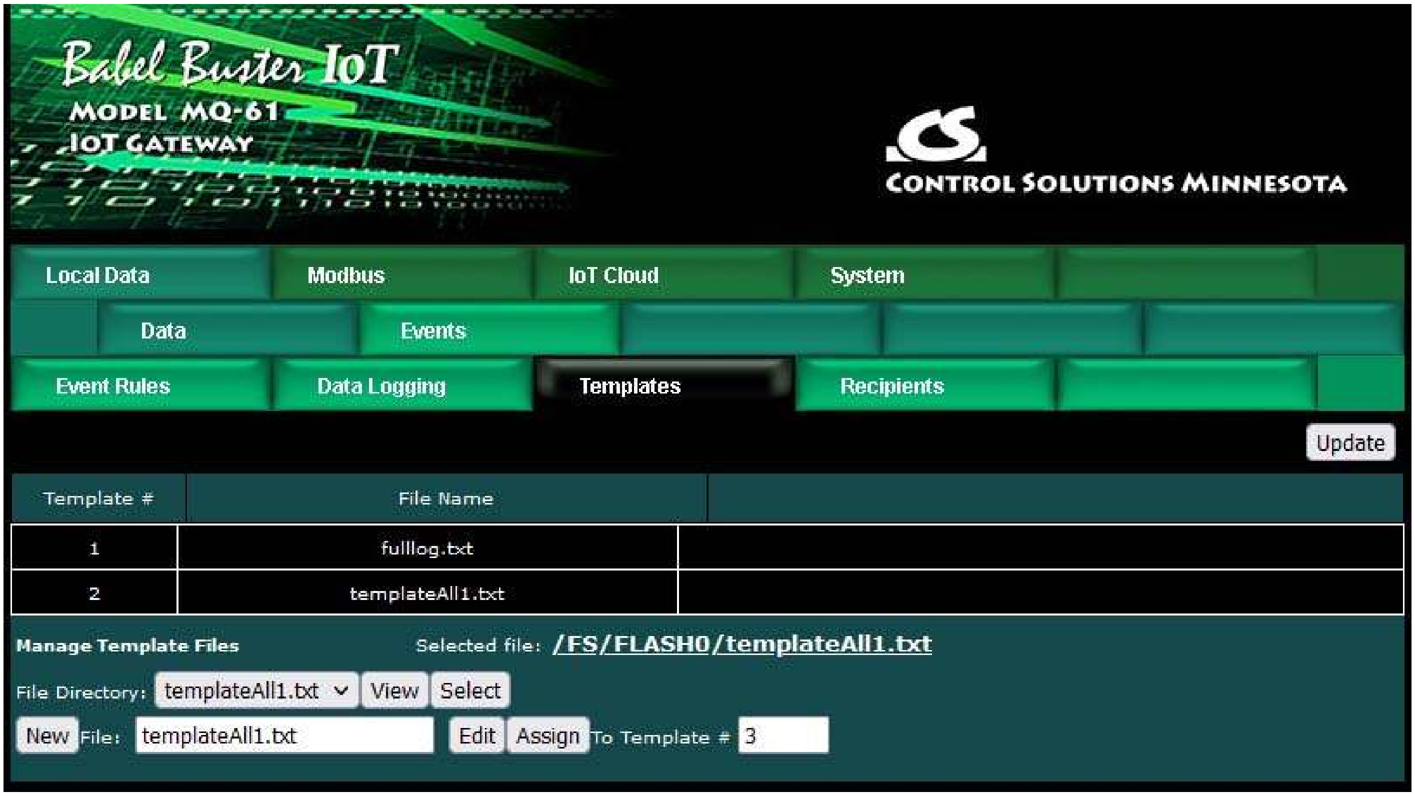
10. Configuring Email Client & Notifications

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: templateAll1.txt View Select

New File: templateAll1.txt Edit Assign To Template # 3

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.

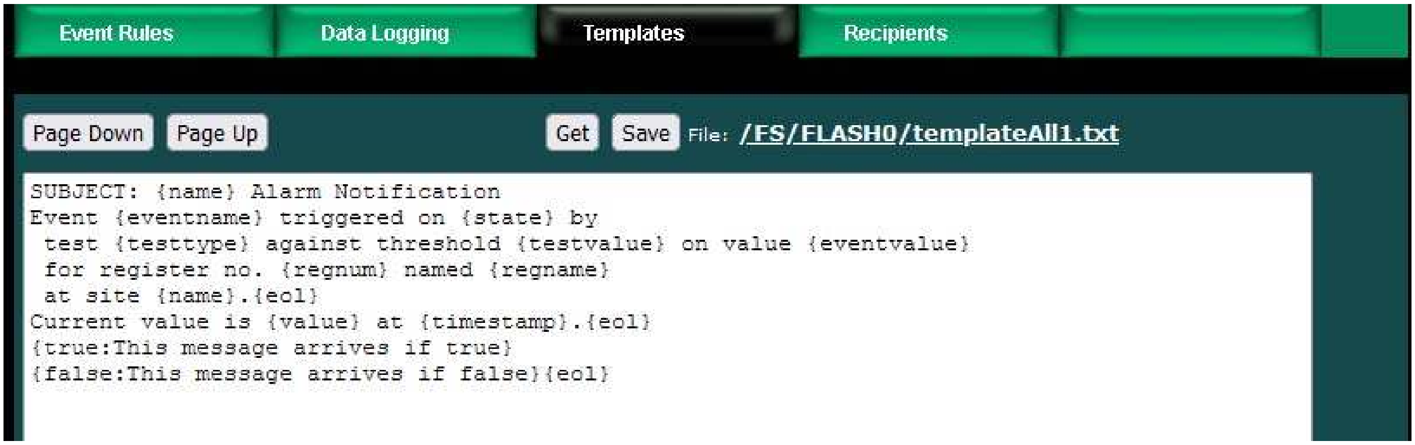
Once you have created a template that shows up in the File Directory list (only *.txt files will be displayed in this list), select a file from the list (pick file from drop-down list, then click Select), enter a number in the "To Template #" window, and then click Assign. The template file itself is 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.

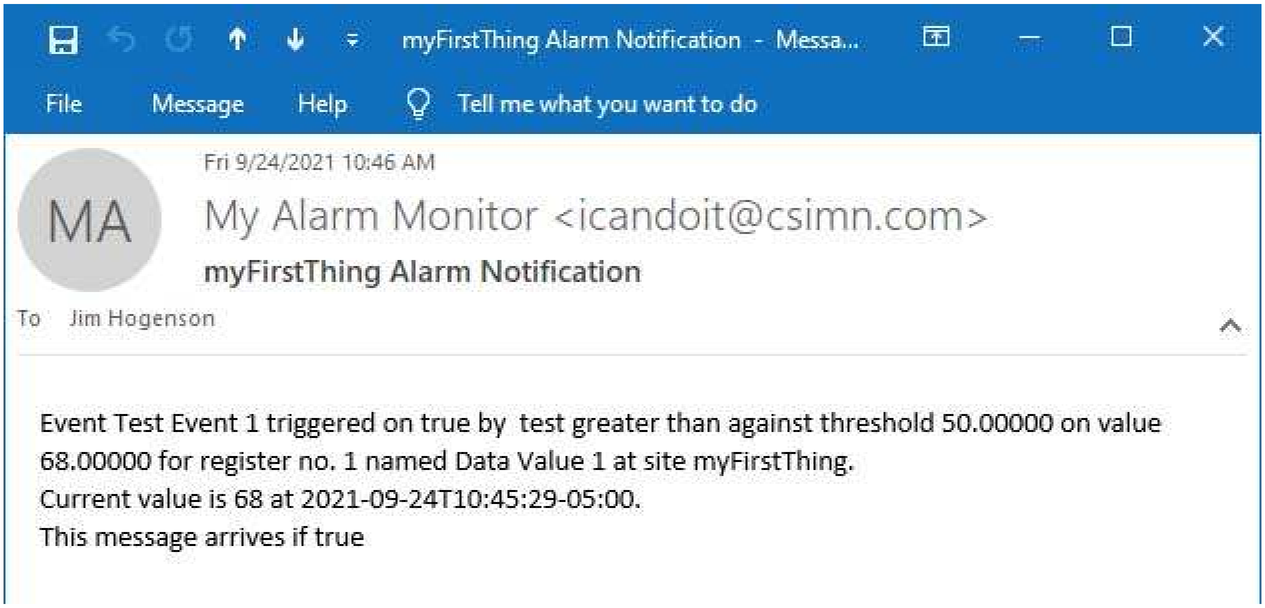
When you have finished editing your template, click Save to save it to the Flash file system.



File: /FS/FLASH0/templateAll1.txt

```
SUBJECT: {name} Alarm Notification
Event {eventname} triggered on {state} by
 test {testtype} against threshold {testvalue} on value {eventvalue}
 for register no. {regnum} named {regname}
 at site {name}.{eol}
Current value is {value} at {timestamp}.{eol}
{true:This message arrives if true}
{false:This message arrives if false}{eol}
```

The above example template produced the following email.


myFirstThing Alarm Notification - Messa...
File
Message
Help
Tell me what you want to do
MA

Fri 9/24/2021 10:46 AM
My Alarm Monitor <icandoit@csimn.com>
myFirstThing Alarm Notification
To Jim Hogenson

Event Test Event 1 triggered on true by test greater than against threshold 50.00000 on value 68.00000 for register no. 1 named Data Value 1 at site myFirstThing.
Current value is 68 at 2021-09-24T10:45:29-05:00.
This message arrives if true

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

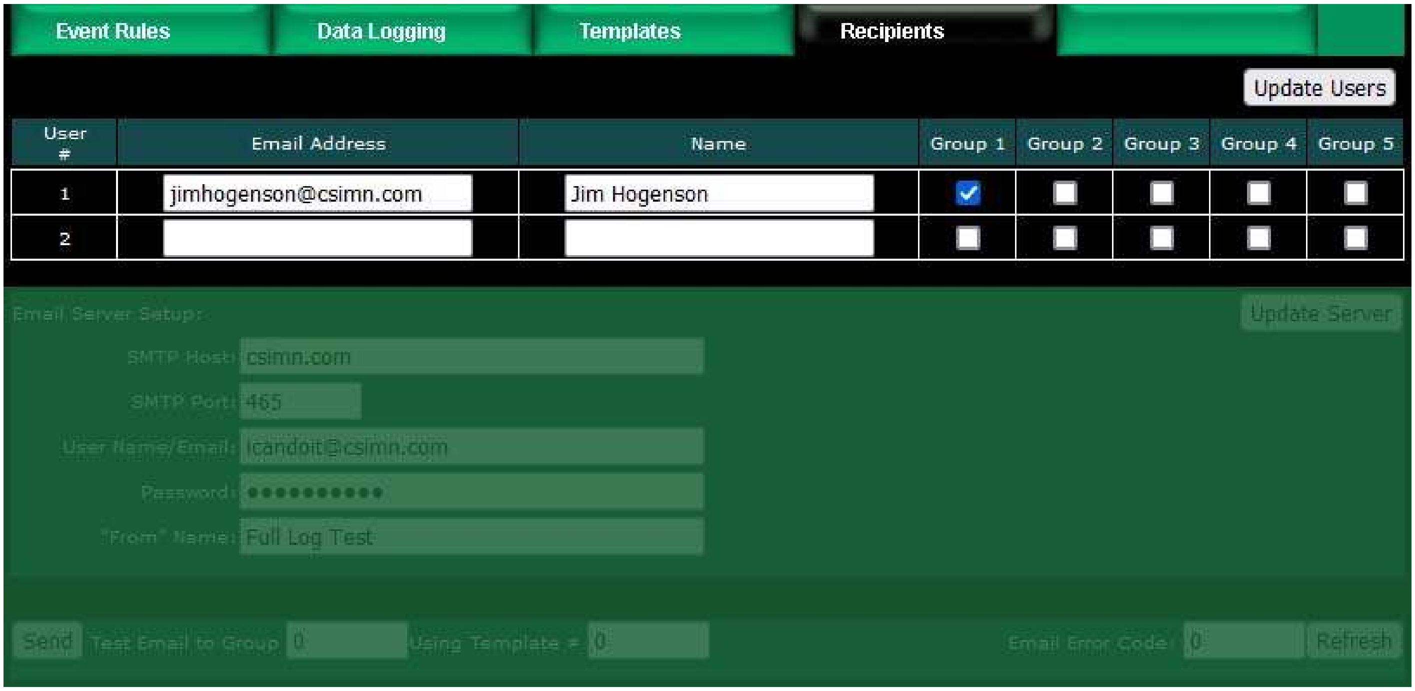
{true:xxx yyy zzz}
Literal string conditional, include "xxx yyy zzz" in message only if event is or transitioned to true

{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	☑	□	□	□	□
2			□	□	□	□	□

Email Server Setup:

SMTP Host: csimn.com
SMTP Port: 465
User Name/Email: icandoit@csimn.com
Password: ●●●●●●●●●●●
"From" Name: Full Log Test

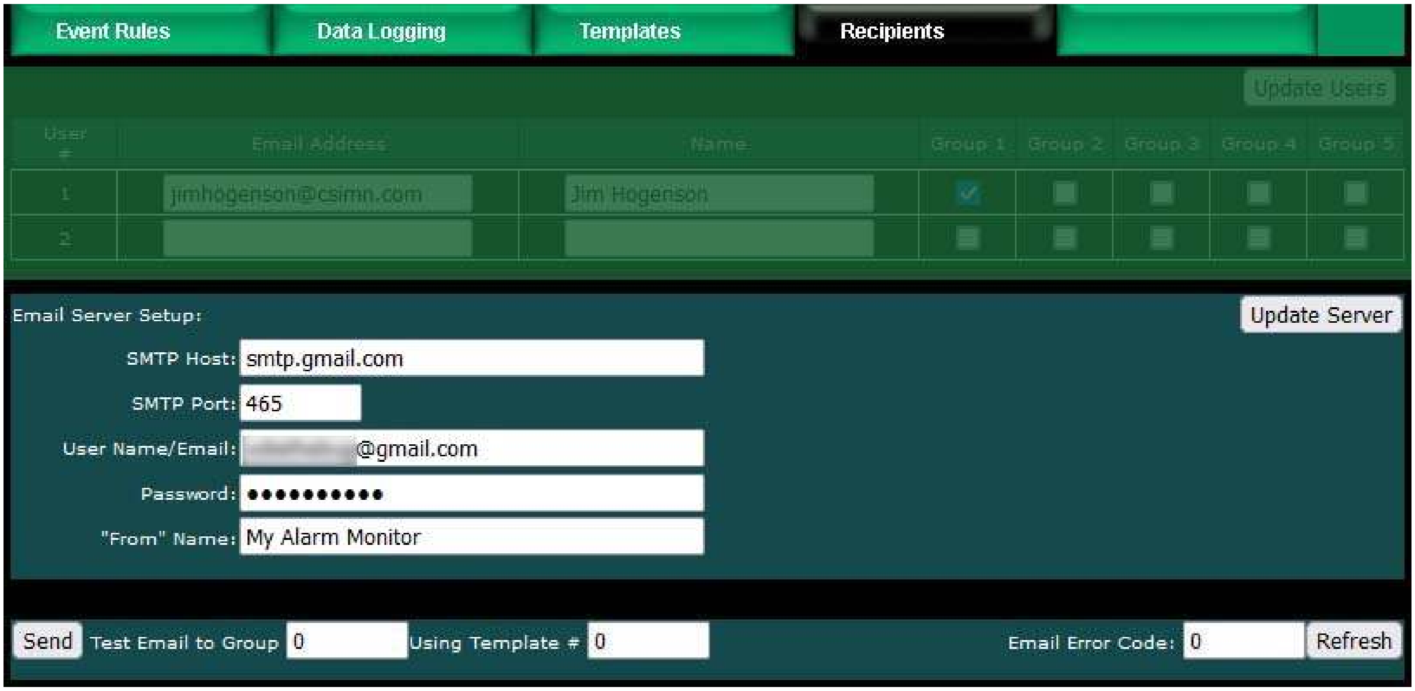
Test Email to Group 0 Using Template # 0

Email Error Code: 0

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	☑	□	□	□	□
2			□	□	□	□	□

Email Server Setup:

SMTP Host: smtp.gmail.com

SMTP Port: 465

User Name/Email: [redacted]@gmail.com

Password: ●●●●●●●●●●

"From" Name: My Alarm Monitor

Send Test Email to Group 0 Using Template # 0 Email Error Code: 0 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

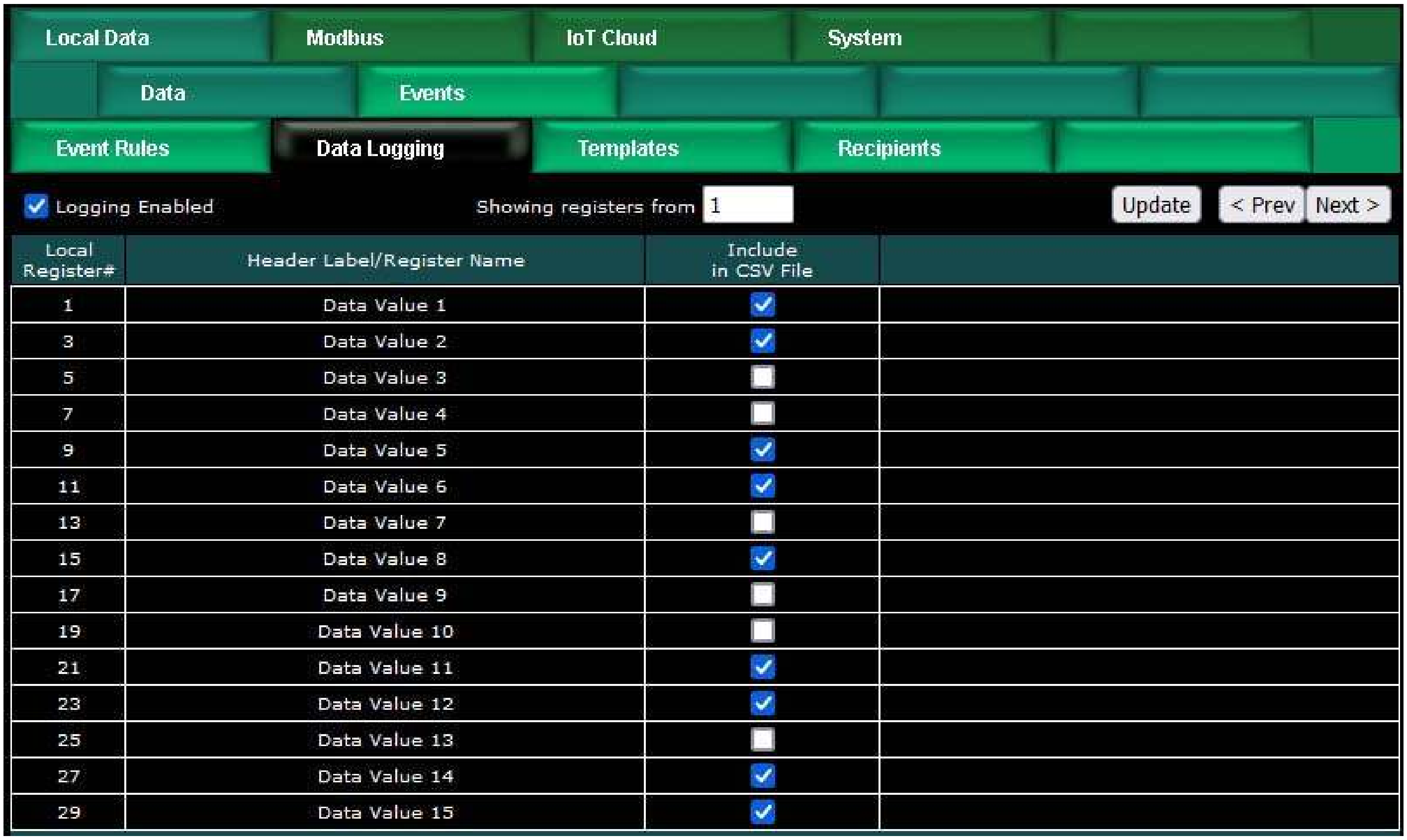
Error codes associated with emails sent by event rules are displayed on the respective event rule page. Error codes associated with emails sent by the data logger are displayed on the data logger page.

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.



☑ Logging Enabled

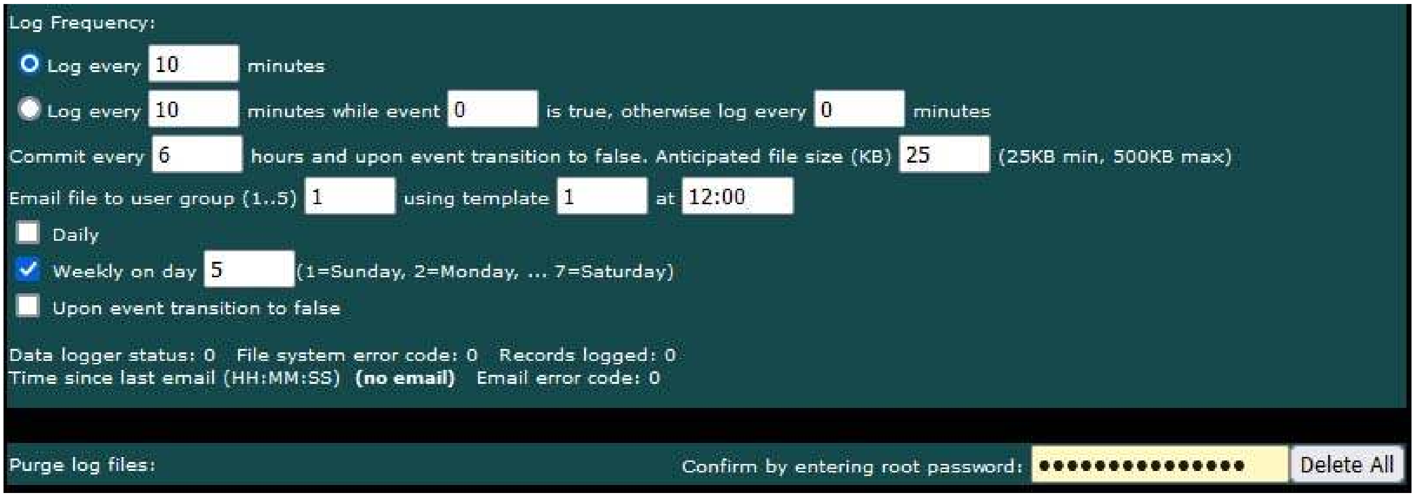
Showing registers from 1

Local Register#	Header Label/Register Name	Include in CSV File
1	Data Value 1	☑
3	Data Value 2	☑
5	Data Value 3	☐
7	Data Value 4	☐
9	Data Value 5	☑
11	Data Value 6	☑
13	Data Value 7	☐
15	Data Value 8	☑
17	Data Value 9	☐
19	Data Value 10	☐
21	Data Value 11	☑
23	Data Value 12	☑
25	Data Value 13	☐
27	Data Value 14	☑
29	Data Value 15	☑

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: ••••••••••••••••

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 spread sheet 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.

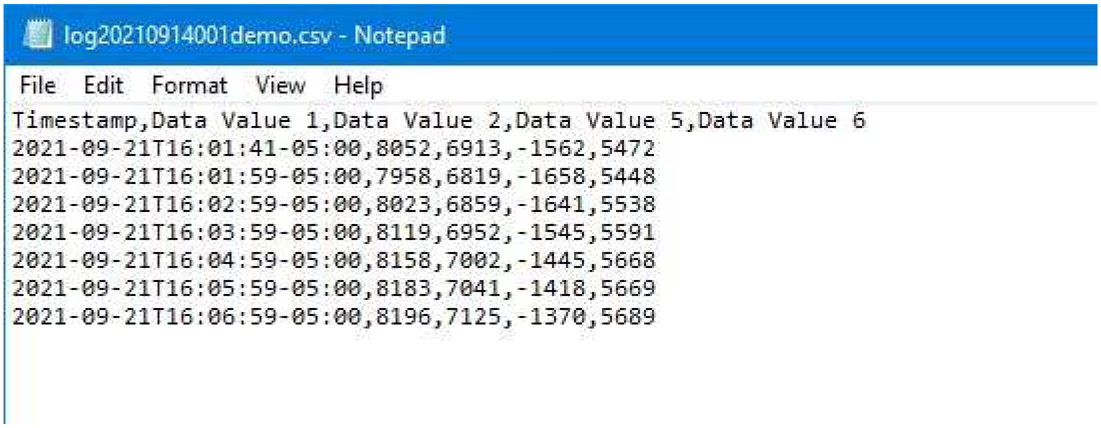
If you have reason to delete all old log files, enter the root password and click Delete All.

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).

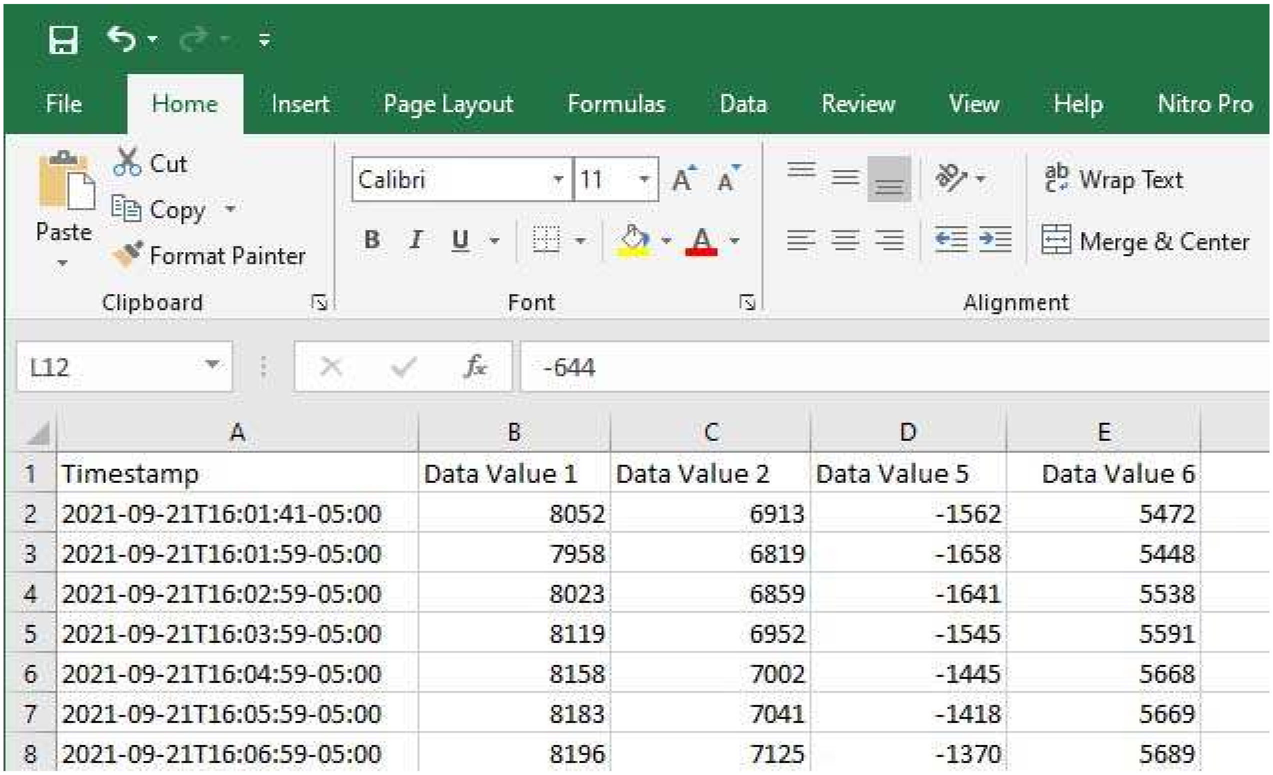
The header line is recorded one time when a new log file is created. Therefore, if you were previously logging data and then change the register selections and promptly resume logging, the data logged now will not correspond to the originally logged header file. To avoid this, retrieve your old log files, 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 snippit of a sample file is illustrated below in raw text form.



```
Timestamp,Data Value 1,Data Value 2,Data Value 5,Data Value 6
2021-09-21T16:01:41-05:00,8052,6913,-1562,5472
2021-09-21T16:01:59-05:00,7958,6819,-1658,5448
2021-09-21T16:02:59-05:00,8023,6859,-1641,5538
2021-09-21T16:03:59-05:00,8119,6952,-1545,5591
2021-09-21T16:04:59-05:00,8158,7002,-1445,5668
2021-09-21T16:05:59-05:00,8183,7041,-1418,5669
2021-09-21T16:06:59-05:00,8196,7125,-1370,5689
```

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



	A	B	C	D	E
1	Timestamp	Data Value 1	Data Value 2	Data Value 5	Data Value 6
2	2021-09-21T16:01:41-05:00	8052	6913	-1562	5472
3	2021-09-21T16:01:59-05:00	7958	6819	-1658	5448
4	2021-09-21T16:02:59-05:00	8023	6859	-1641	5538
5	2021-09-21T16:03:59-05:00	8119	6952	-1545	5591
6	2021-09-21T16:04:59-05:00	8158	7002	-1445	5668
7	2021-09-21T16:05:59-05:00	8183	7041	-1418	5669
8	2021-09-21T16:06:59-05:00	8196	7125	-1370	5689

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 IoT Gateway 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.

Babel Buster IoT
MODEL MQ-61
MODEL BB3-6101-MQ
IOT GATEWAY

CONTROL SOLUTIONS MINNESOTA

12. Configuring the Scheduler

The Babel Buster IoT Gateway 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.

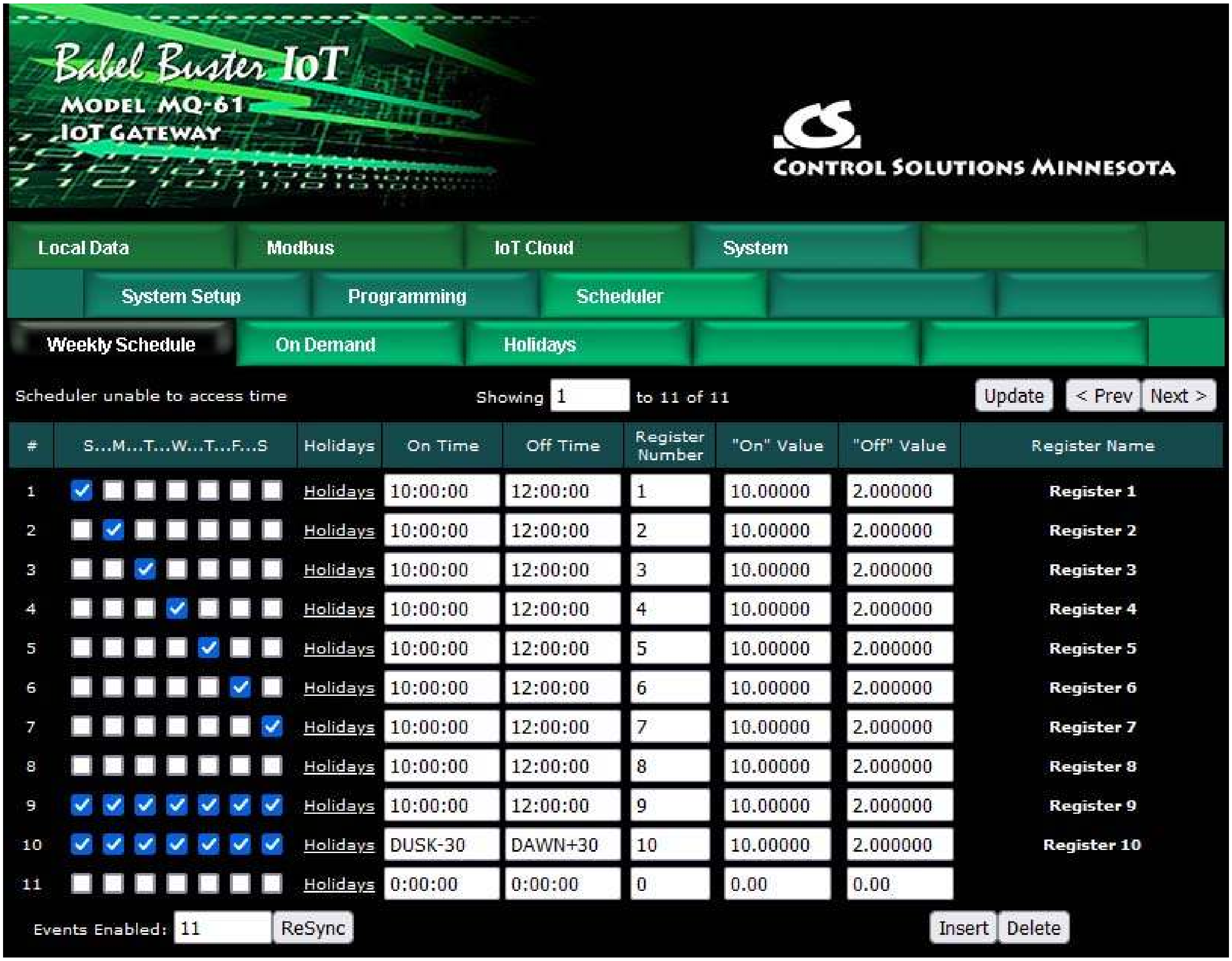
12.1 Weekly Schedule

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 1 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.



Scheduler unable to access time

Showing 1 to 11 of 11

#	S...M...T...W...T...F...S	Holidays	On Time	Off Time	Register Number	"On" Value	"Off" Value	Register Name
1	☑ ☐ ☐ ☐ ☐ ☐ ☐	Holidays	10:00:00	12:00:00	1	10.00000	2.000000	Register 1
2	☐ ☑ ☐ ☐ ☐ ☐ ☐	Holidays	10:00:00	12:00:00	2	10.00000	2.000000	Register 2
3	☐ ☐ ☑ ☐ ☐ ☐ ☐	Holidays	10:00:00	12:00:00	3	10.00000	2.000000	Register 3
4	☐ ☐ ☐ ☑ ☐ ☐ ☐	Holidays	10:00:00	12:00:00	4	10.00000	2.000000	Register 4
5	☐ ☐ ☐ ☐ ☑ ☐ ☐	Holidays	10:00:00	12:00:00	5	10.00000	2.000000	Register 5
6	☐ ☐ ☐ ☐ ☐ ☑ ☐	Holidays	10:00:00	12:00:00	6	10.00000	2.000000	Register 6
7	☐ ☐ ☐ ☐ ☐ ☐ ☑	Holidays	10:00:00	12:00:00	7	10.00000	2.000000	Register 7
8	☐ ☐ ☐ ☐ ☐ ☐ ☐	Holidays	10:00:00	12:00:00	8	10.00000	2.000000	Register 8
9	☑ ☑ ☑ ☑ ☑ ☑ ☑	Holidays	10:00:00	12:00:00	9	10.00000	2.000000	Register 9
10	☑ ☑ ☑ ☑ ☑ ☑ ☑	Holidays	DUSK-30	DAWN+30	10	10.00000	2.000000	Register 10
11	☐ ☐ ☐ ☐ ☐ ☐ ☐	Holidays	0:00:00	0:00:00	0	0.00	0.00	

Events Enabled: 11

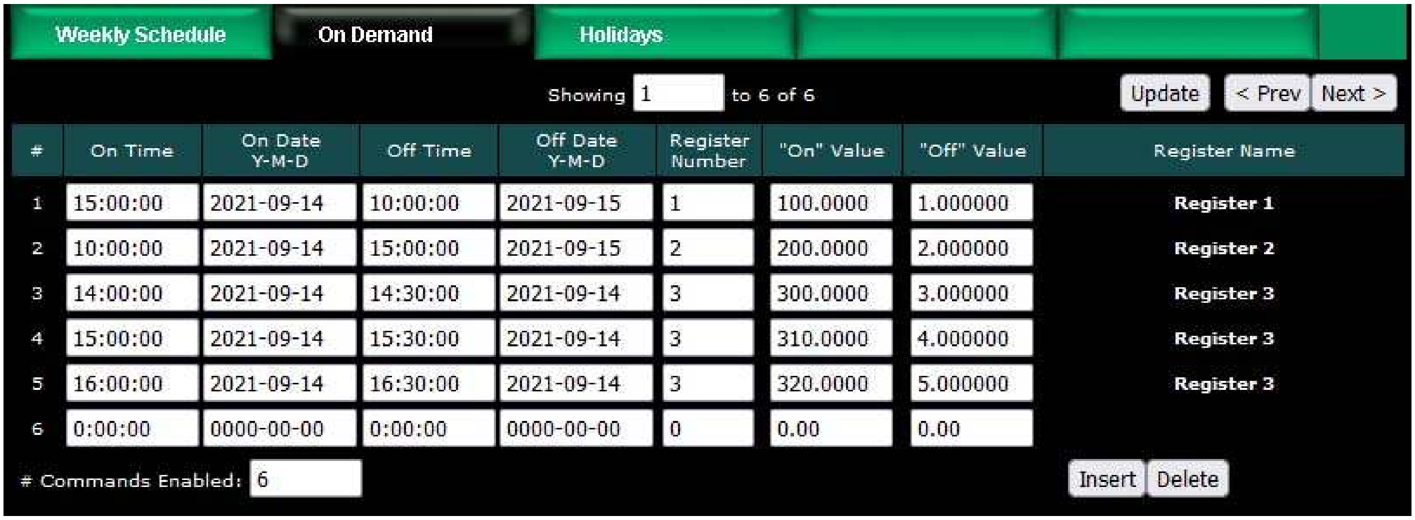
Note the ReSync button at the bottom. If you have made changes to the scheduler, ReSync will cause everything about the schedule to be re-evaluated and registers updated accordingly. Normally, registers are written only when the schedule says it is time to change. The evaluation is made at the start 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 1 will be set to a value of 100 starting at 3:00PM September 14, 2021, and remain at that value until 10:00AM September 15. At all other times, local regiseter 1 will be set to 1. And so forth.



Showing 1 to 6 of 6

#	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	2021-09-14	10:00:00	2021-09-15	1	100.0000	1.000000	Register 1
2	10:00:00	2021-09-14	15:00:00	2021-09-15	2	200.0000	2.000000	Register 2
3	14:00:00	2021-09-14	14:30:00	2021-09-14	3	300.0000	3.000000	Register 3
4	15:00:00	2021-09-14	15:30:00	2021-09-14	3	310.0000	4.000000	Register 3
5	16:00:00	2021-09-14	16:30:00	2021-09-14	3	320.0000	5.000000	Register 3
6	0:00:00	0000-00-00	0:00:00	0000-00-00	0	0.00	0.00	

Commands Enabled: 6

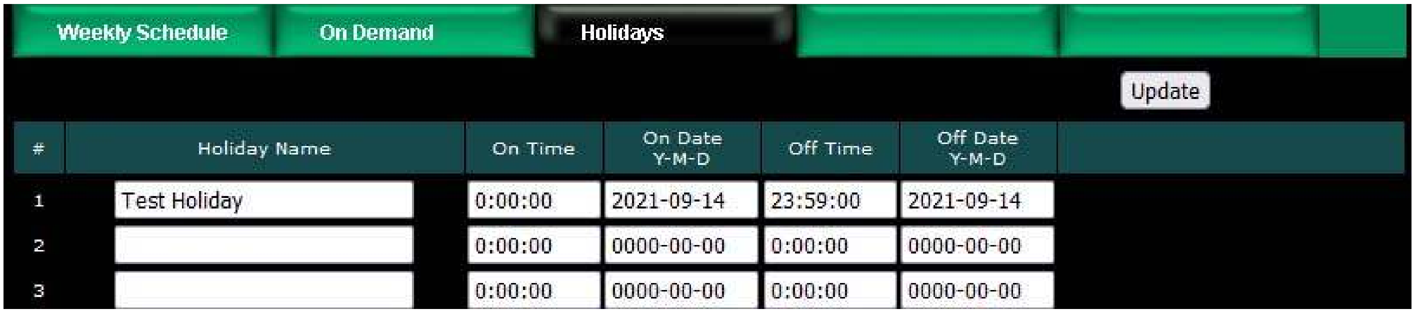
Entries applied to the same register number will be processed sequentially. Register 3 in the above example will be set to a value of 3 prior to 2:00PM Sept. 14. Then from 2:00PM to 2:30PM on Sept. 14, it will be set to a value of 300. From 2:30PM to 3:00PM, the value will be 3. From 3:00PM to 3:30PM, 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 Sept. 14, the value will be 5. On October 1, the value in register 3 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

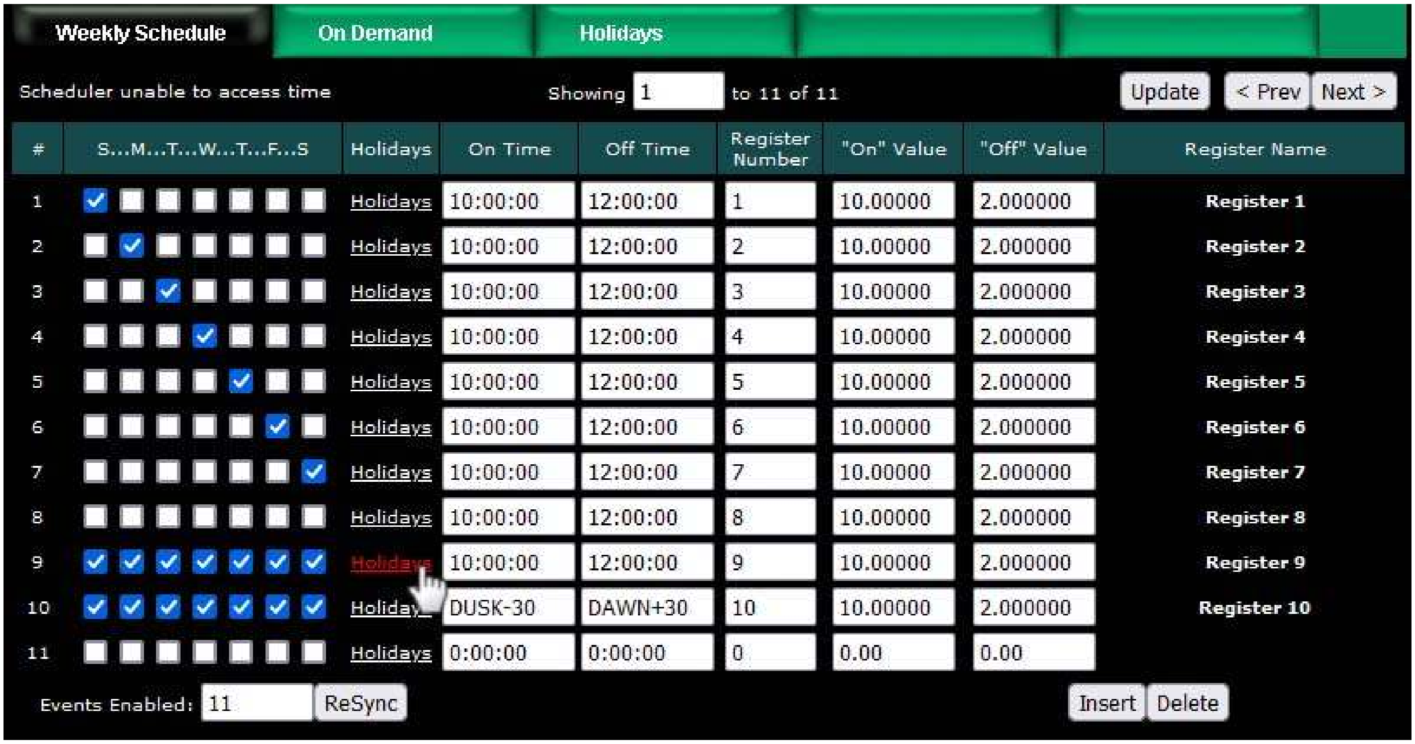
Sometimes you want a weekly schedule to not apply on a holiday, or maybe you want something to only happen on a holiday (although that would be nearly the same as On Demand). The holiday processing in the scheduler allows exceptions to the weekly schedule.

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.



#	Holiday Name	On Time	On Date Y-M-D	Off Time	Off Date Y-M-D
1	Test Holiday	0:00:00	2021-09-14	23:59:00	2021-09-14
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.



Scheduler unable to access time

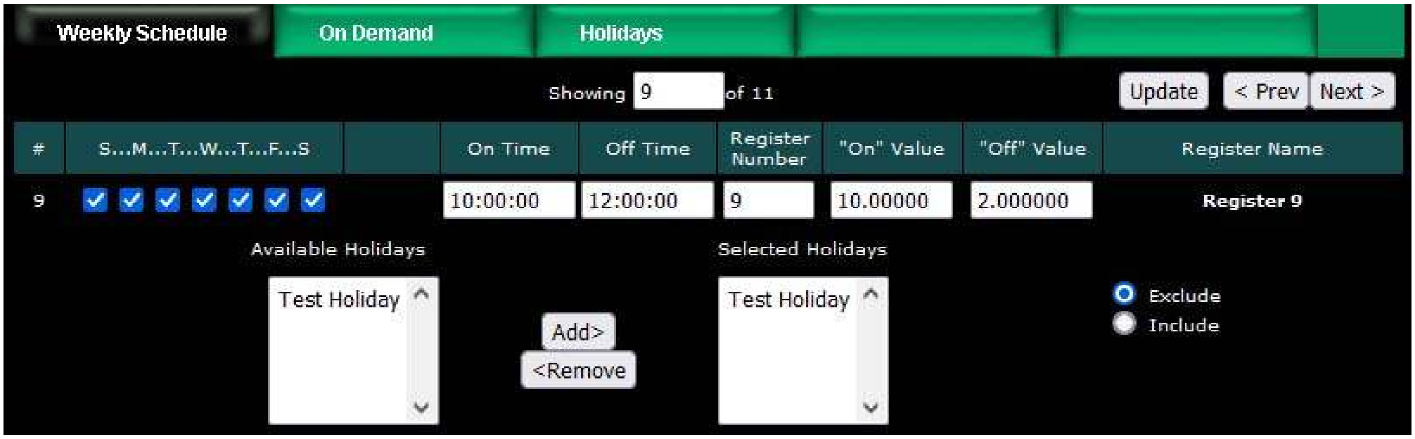
Showing 1 to 11 of 11

#	S...M...T...W...T...F...S	Holidays	On Time	Off Time	Register Number	"On" Value	"Off" Value	Register Name
1	☑ □ □ □ □ □ □	Holidays	10:00:00	12:00:00	1	10.00000	2.000000	Register 1
2	□ ☑ □ □ □ □ □	Holidays	10:00:00	12:00:00	2	10.00000	2.000000	Register 2
3	□ □ ☑ □ □ □ □	Holidays	10:00:00	12:00:00	3	10.00000	2.000000	Register 3
4	□ □ □ ☑ □ □ □	Holidays	10:00:00	12:00:00	4	10.00000	2.000000	Register 4
5	□ □ □ □ ☑ □ □	Holidays	10:00:00	12:00:00	5	10.00000	2.000000	Register 5
6	□ □ □ □ □ ☑ □	Holidays	10:00:00	12:00:00	6	10.00000	2.000000	Register 6
7	□ □ □ □ □ □ ☑	Holidays	10:00:00	12:00:00	7	10.00000	2.000000	Register 7
8	□ □ □ □ □ □ □	Holidays	10:00:00	12:00:00	8	10.00000	2.000000	Register 8
9	☑ ☑ ☑ ☑ ☑ ☑ ☑	Holidays	10:00:00	12:00:00	9	10.00000	2.000000	Register 9
10	☑ ☑ ☑ ☑ ☑ ☑ ☑	Holidays	DUSK-30	DAWN+30	10	10.00000	2.000000	Register 10
11	□ □ □ □ □ □ □	Holidays	0:00:00	0:00:00	0	0.00	0.00	

Events Enabled: 11

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.



Showing 9 of 11

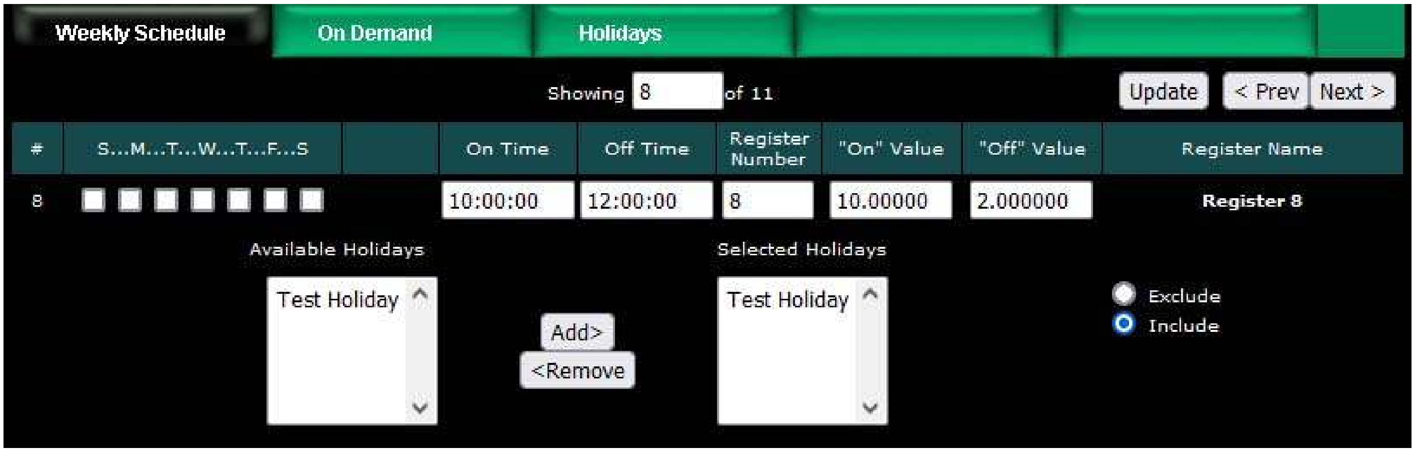
#	S...M...T...W...T...F...S		On Time	Off Time	Register Number	"On" Value	"Off" Value	Register Name
9	☑ ☑ ☑ ☑ ☑ ☑ ☑		10:00:00	12:00:00	9	10.00000	2.000000	Register 9

Available Holidays: Test Holiday

Selected Holidays: Test Holiday

● Exclude
○ Include

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.)



Showing 8 of 11

#	S...M...T...W...T...F...S		On Time	Off Time	Register Number	"On" Value	"Off" Value	Register Name
8	□ □ □ □ □ □ □		10:00:00	12:00:00	8	10.00000	2.000000	Register 8

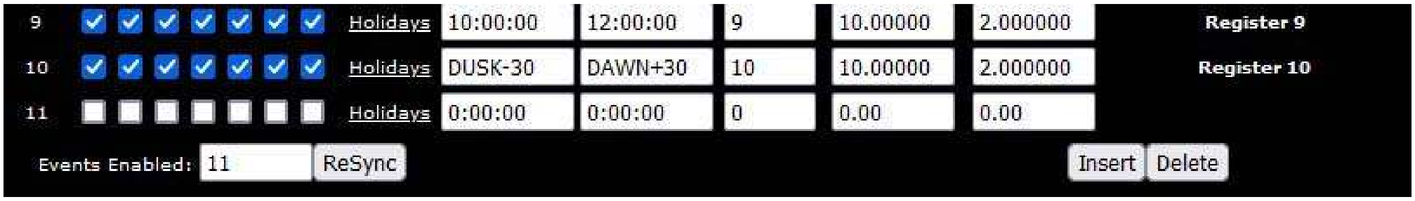
Available Holidays: Test Holiday

Selected Holidays: Test Holiday

○ Exclude
● Include

12.4 Astronomical Clock

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



9	☑ ☑ ☑ ☑ ☑ ☑ ☑	Holidays	10:00:00	12:00:00	9	10.00000	2.000000	Register 9
10	☑ ☑ ☑ ☑ ☑ ☑ ☑	Holidays	DUSK-30	DAWN+30	10	10.00000	2.000000	Register 10
11	□ □ □ □ □ □ □	Holidays	0:00:00	0:00:00	0	0.00	0.00	

Events Enabled: 11

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 10 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.

Babel Buster IoT
MODEL MQ-61
MODEL BB3-6101-MQ
IOT GATEWAY

CONTROL SOLUTIONS MINNESOTA

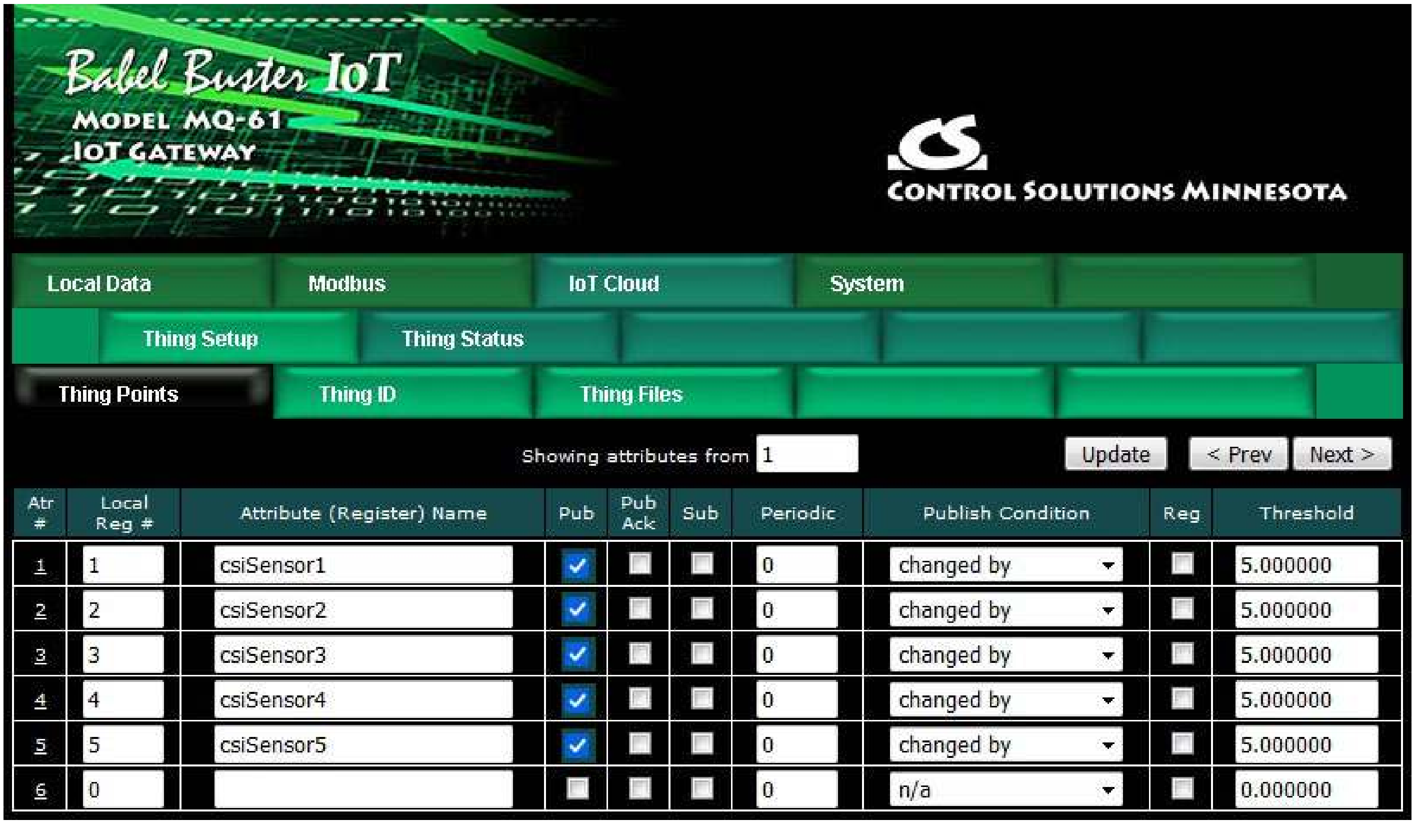
13. Configuring the IoT Gateway

The pages used to configure the Babel Buster IoT Gateway'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 Gateway and the AWS IoT features, you will want to refer to Sections 14 through 18 of this user guide.

13.1 Thing Points or Attributes

The Babel Buster IoT Gateway is used to turn any Modbus device into a "thing" 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 IoT Gateway maps Modbus registers 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.

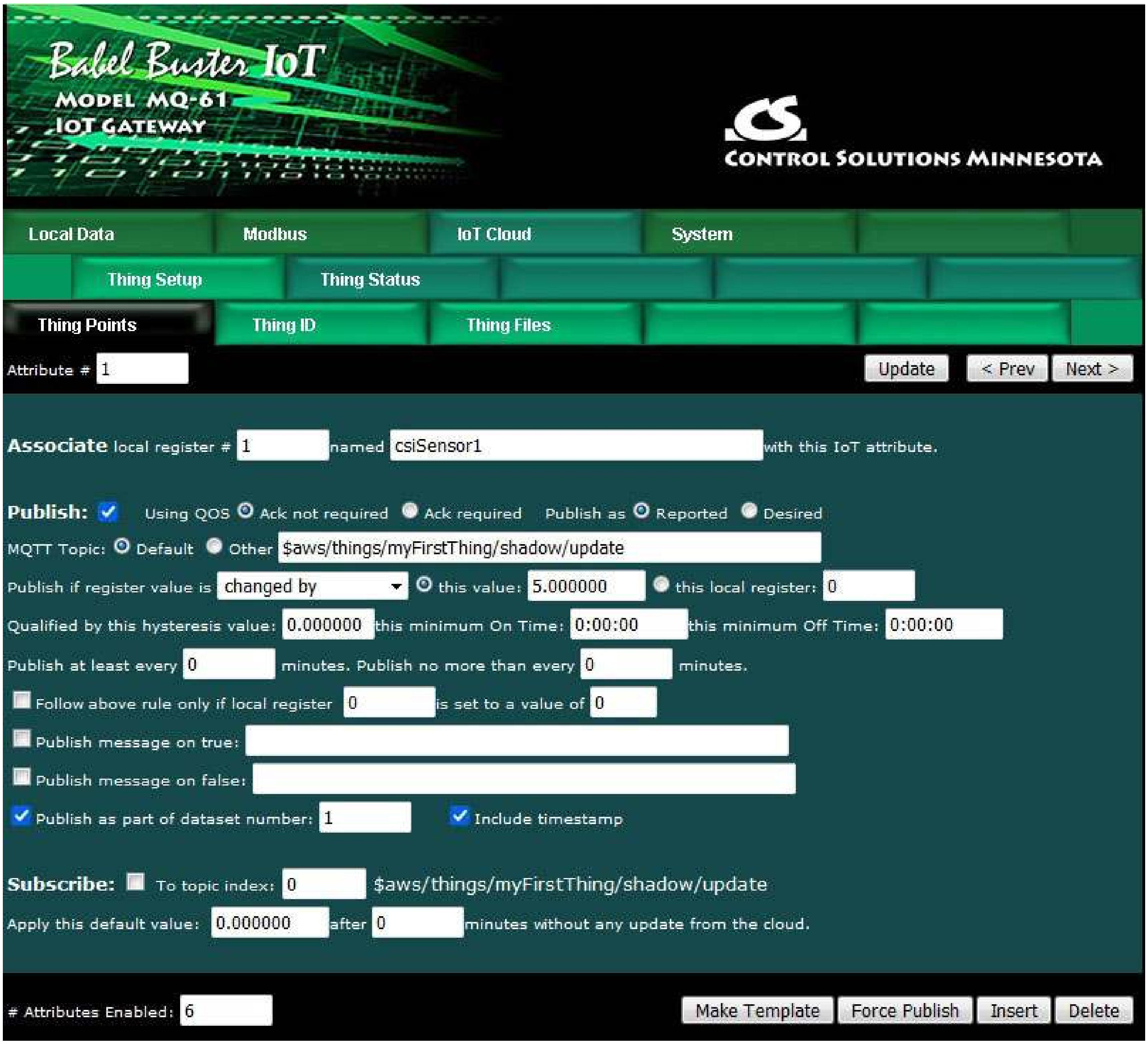
The Thing Points page shows a list of all current attributes, i.e., local registers that have been mapped as an attribute for our "thing". You can create very simple publish and subscribe rules on this tabular list of attributes, but for maximum flexibility, you will want to review each attribute individually.



Showing attributes from 1

Atr #	Local Reg #	Attribute (Register) Name	Pub	Pub Ack	Sub	Periodic	Publish Condition	Reg	Threshold
1	1	csiSensor1	☑	□	□	0	changed by	□	5.000000
2	2	csiSensor2	☑	□	□	0	changed by	□	5.000000
3	3	csiSensor3	☑	□	□	0	changed by	□	5.000000
4	4	csiSensor4	☑	□	□	0	changed by	□	5.000000
5	5	csiSensor5	☑	□	□	0	changed by	□	5.000000
6	0		□	□	□	0	n/a	□	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.



Attribute # 1

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 $aws/things/myFirstThing/shadow/update

Publish if register value is changed by ● this value: 5.000000 ○ 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

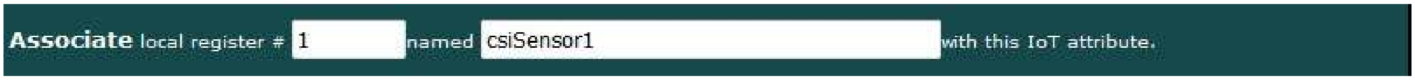
Subscribe: □ To topic index: 0 $aws/things/myFirstThing/shadow/update

Apply this default value: 0.000000 after 0 minutes without any update from the cloud.

Attributes Enabled: 6

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.

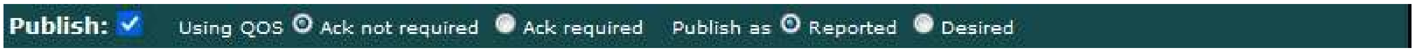


Associate local register # 1 named csiSensor1 with this IoT attribute.

Select Publish if you wish to publish this point. Publish means send data from the IoT Gateway to the 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 IoT Gateway will repeatedly retry publishing until the server responds with an acknowledgement.

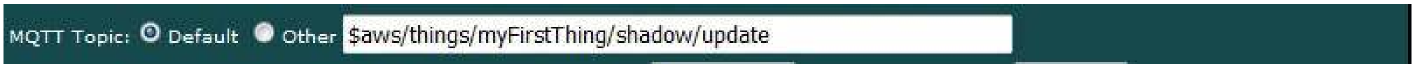
You will normally publish data points as "Reported". If, however, you are publishing to the shadow object of another IoT Gateway or similar IoT device, and your intent is to set a register value in that remote device via the AWS server, then you would publish as "Desired".



Publish: ☑ Using QOS ● Ack not required ○ Ack required Publish as ● Reported ○ 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 IoT Gateway that you will set up on the AWS server (see Section 14).

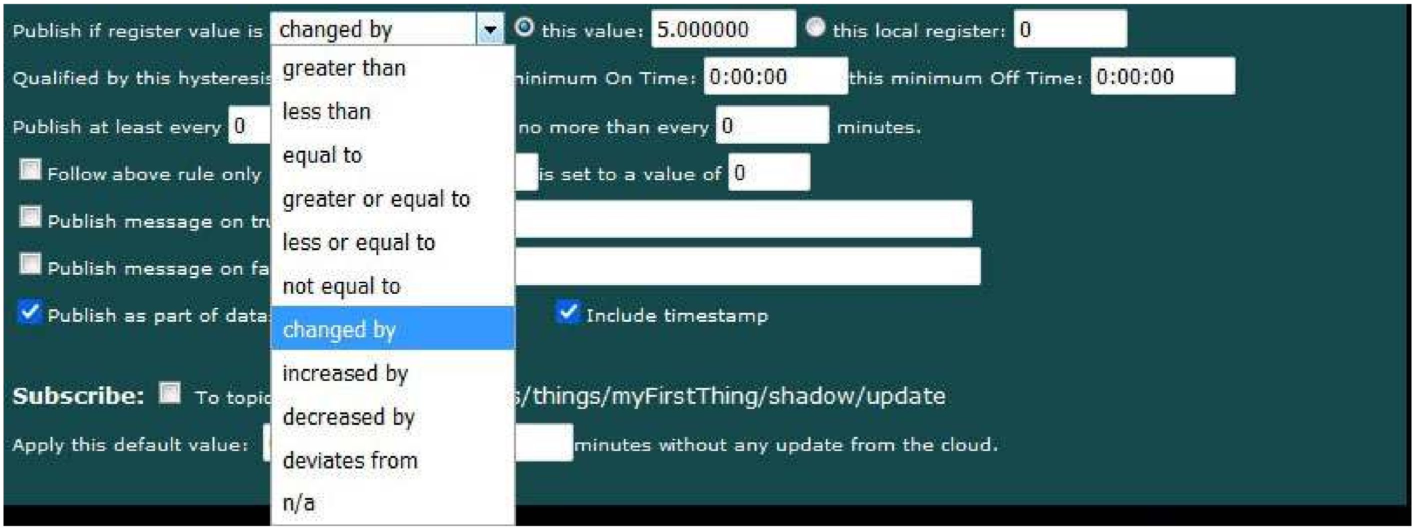
If you wish to publish to a topic other than the default topic, enter that topic here and select "Other".



MQTT Topic: ● Default ○ Other $aws/things/myFirstThing/shadow/update

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.



Publish if register value is changed by ● this value: 5.000000 ○ this local register: 0

Qualified by this hysteresis inimum On Time: 0:00:00 this minimum Off Time: 0:00:00

Publish at least every 0 no more than every 0 minutes.

□ Follow above rule only is set to a value of 0

□ Publish message on tr

□ Publish message on fa

☑ Publish as part of data ☑ Include timestamp

Subscribe: □ To topic s/things/myFirstThing/shadow/update

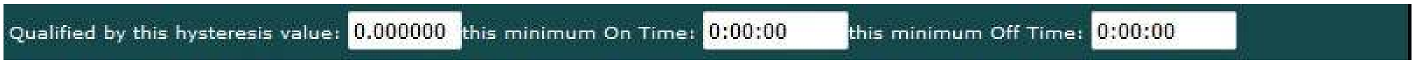
Apply this default value: minutes without any update from the cloud.

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.

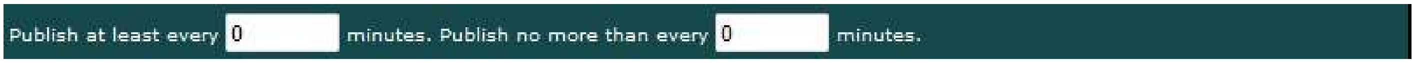
On time and off time, if specified, determine how long the condition must be true (on time) or false (off time) before the true or false response is actually taken. Times are given in HH:MM:SS format (hours, minutes, seconds). On/off time qualifications should not be used with "changed by" or other transition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



Qualified by this hysteresis value: 0.000000 this minimum On Time: 0:00:00 this minimum Off Time: 0:00:00

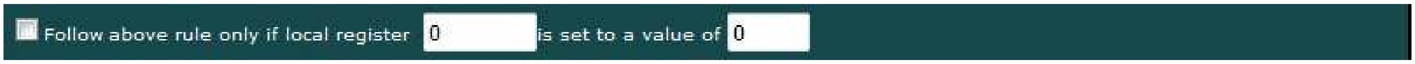
Publish at least every N minutes will result in periodic publishing regardless of any conditional testing. Periodic publishing is disabled by entering zero here.

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.



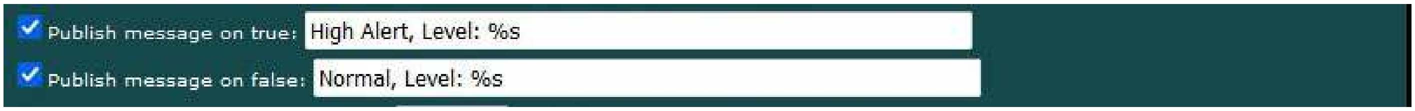
Publish at least every 0 minutes. Publish no more than every 0 minutes.

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 0 is set to a value of 0

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.

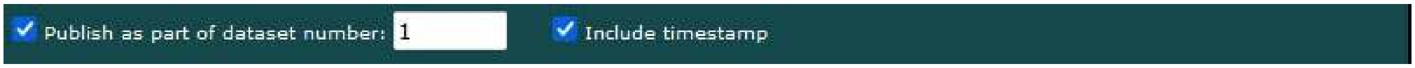


☑ Publish message on true: High Alert, Level: %s

☑ Publish message on false: Normal, Level: %s

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".



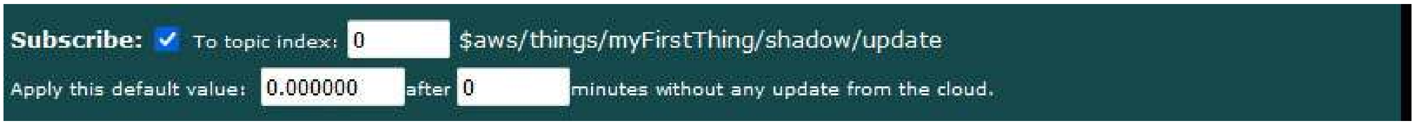
☑ Publish as part of dataset number: 1 ☑ Include timestamp

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.

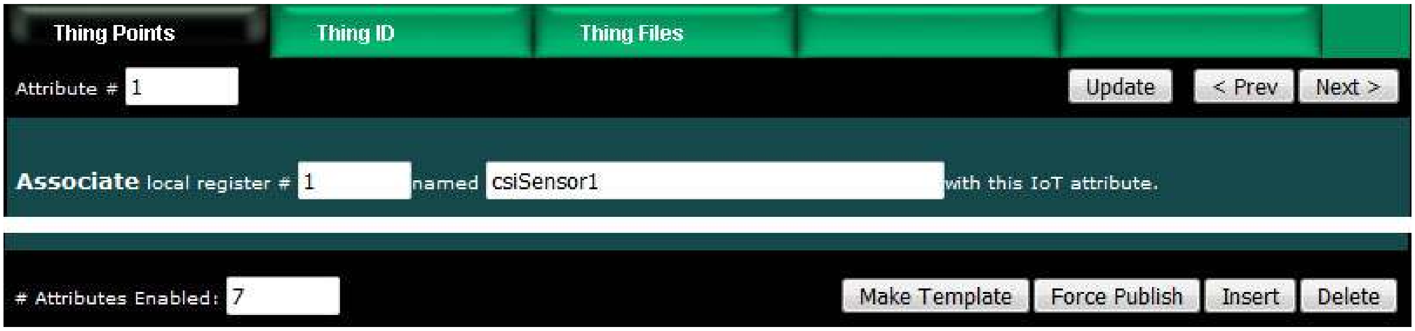


Subscribe: ☑ To topic index: 0 $aws/things/myFirstThing/shadow/update

Apply this default value: 0.000000 after 0 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 IoT Gateway. 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.



Attribute # 1

Associate local register # 1 named csiSensor1 with this IoT attribute.

Attributes Enabled: 7

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 spread sheet. 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.

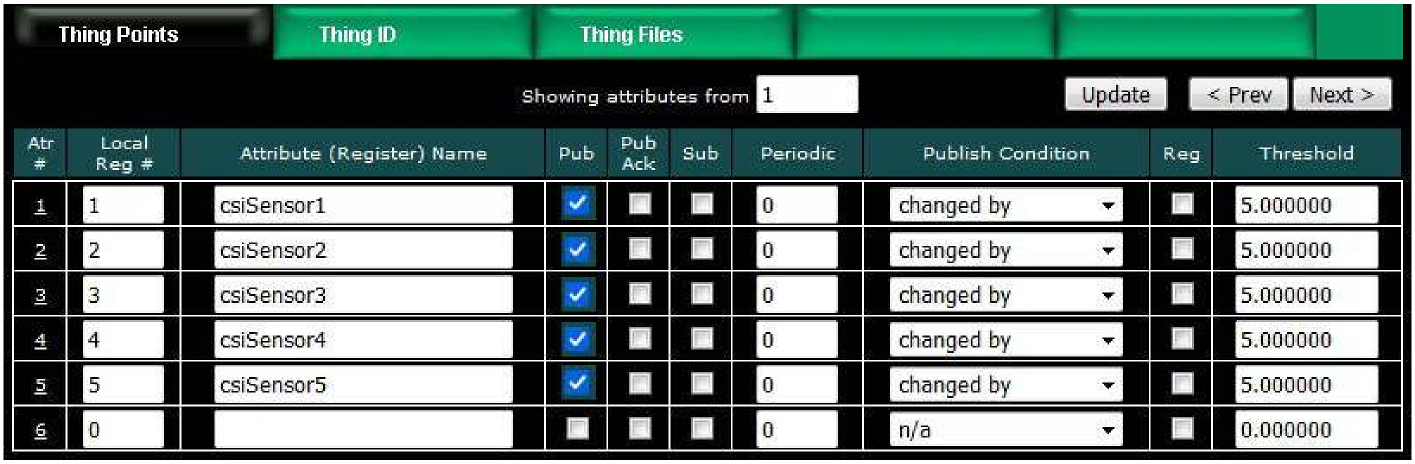
Delete will remove the attribute number shown in the "Showing" box. Entering zero as the "Associate" register will also effectively deletes the attribute even though it will still appear in the list until deleted. Unused attributes at the end of the list will always show zero as the associate register. 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.



Showing attributes from 1

Atr #	Local Reg #	Attribute (Register) Name	Pub	Pub Ack	Sub	Periodic	Publish Condition	Reg	Threshold
1	1	csiSensor1	☑	□	□	0	changed by	□	5.000000
2	2	csiSensor2	☑	□	□	0	changed by	□	5.000000
3	3	csiSensor3	☑	□	□	0	changed by	□	5.000000
4	4	csiSensor4	☑	□	□	0	changed by	□	5.000000
5	5	csiSensor5	☑	□	□	0	changed by	□	5.000000
6	0		□	□	□	0	n/a	□	0.000000

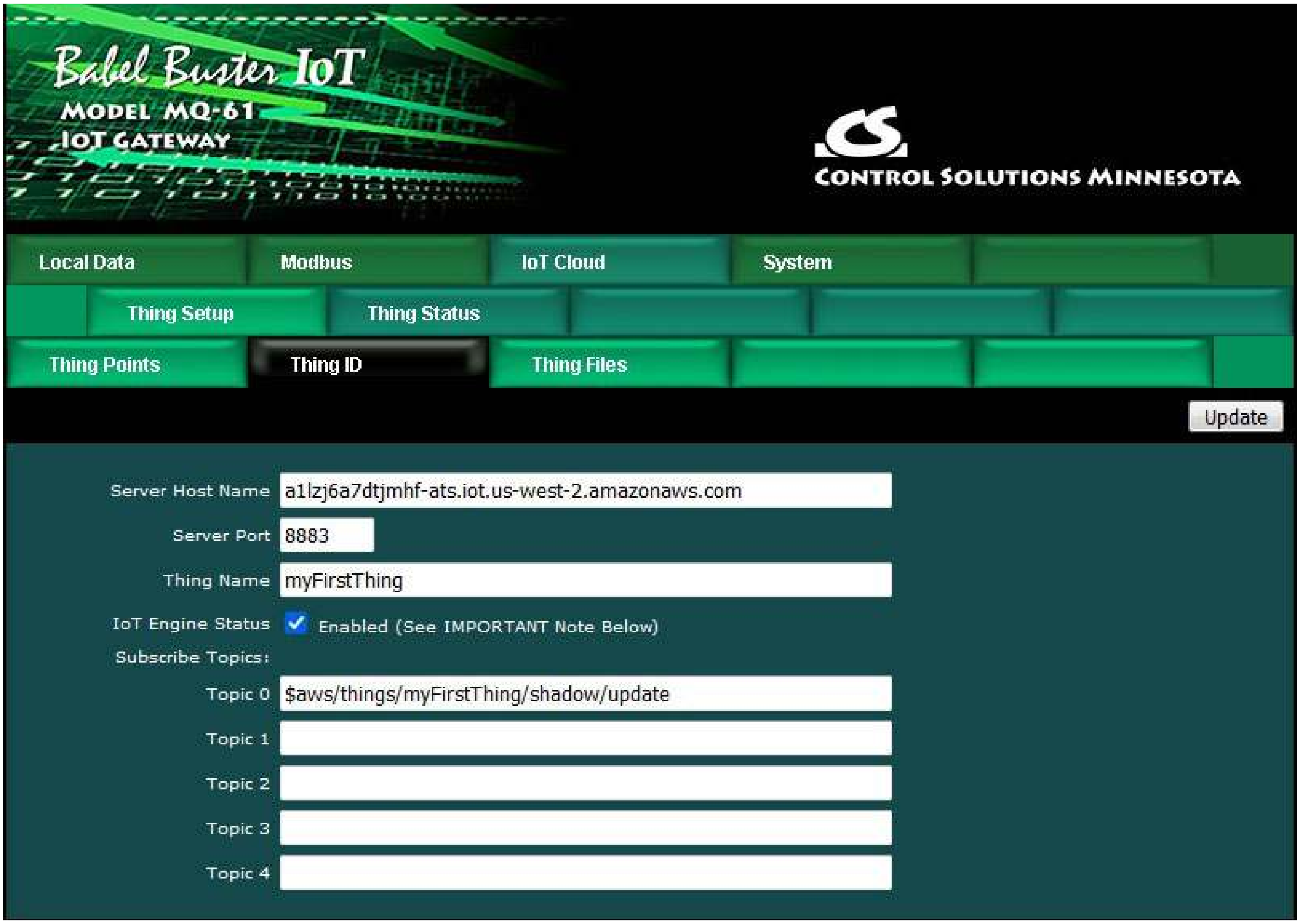
Referring to the preceding detailed descriptions, select a local register number to be published or subscribed, and allow the IoT Gateway to retrieve the name for you or enter a new name following the guidelines noted above.

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 IoT Gateway's connection to the AWS server. The Subscribe Topics are used if subscribing, and are not used to publish.



Server Host Name a1lzj6a7dtjmhf-ats.iot.us-west-2.amazonaws.com

Server Port 8883

Thing Name myFirstThing

IoT Engine Status ☑ Enabled (See IMPORTANT Note Below)

Subscribe Topics:

Topic 0 $aws/things/myFirstThing/shadow/update

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 IoT Gateway 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 IoT Gateway 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.

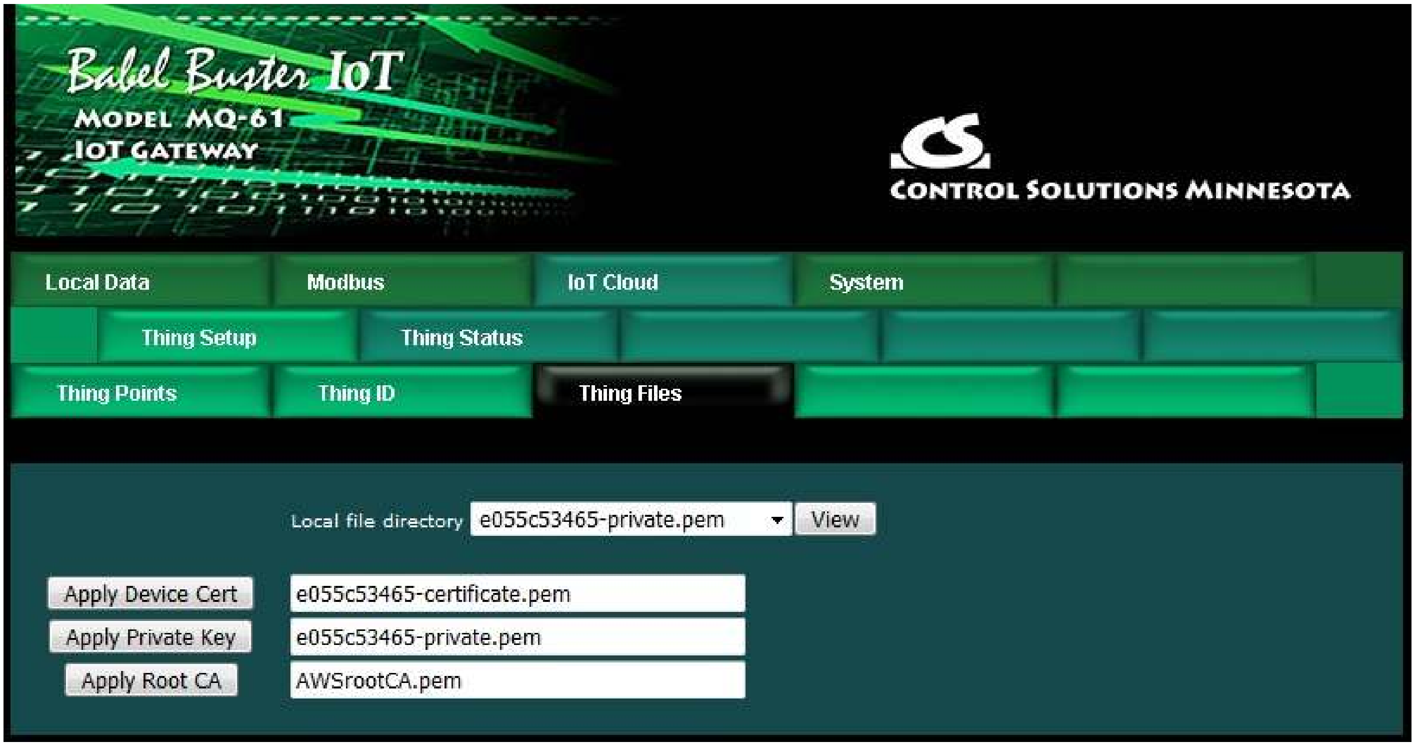
DO NOT ENABLE the IoT Engine with invalid configuration information. Doing so for an extended period may get you blocked by the Amazon servers. If you see errors reported by Amazon on the Thing Status page, it is a good idea to disable the IoT Engine while you figure out why the error(s) occurred.

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 IoT Gateway and the AWS server is a secure connection requiring valid SSL certificates.



Local file directory e055c53465-private.pem

Apply Device Cert e055c53465-certificate.pem

Apply Private Key e055c53465-private.pem

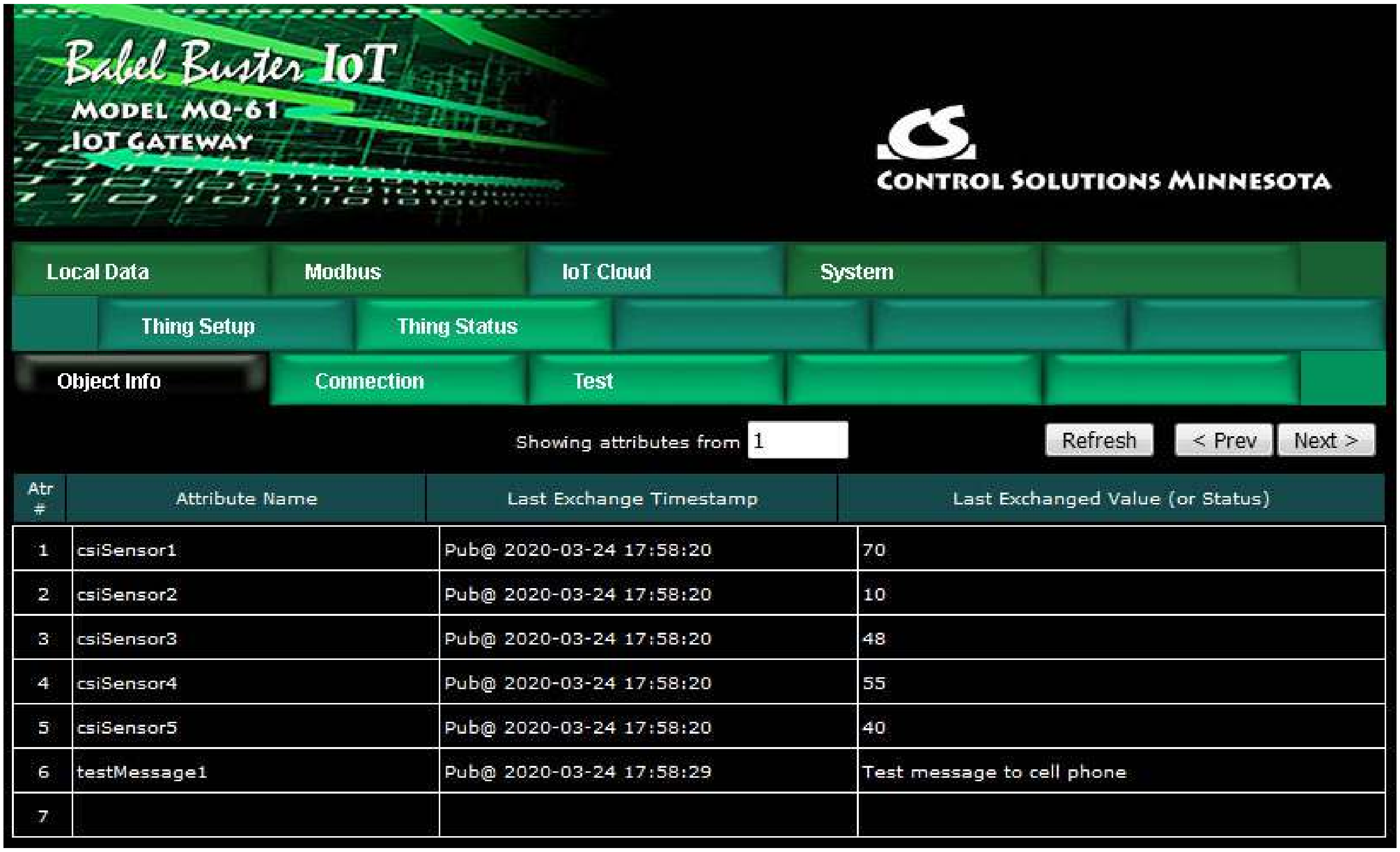
Apply Root CA AWSrootCA.pem

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 IoT Gateway.

13.4 Thing Status

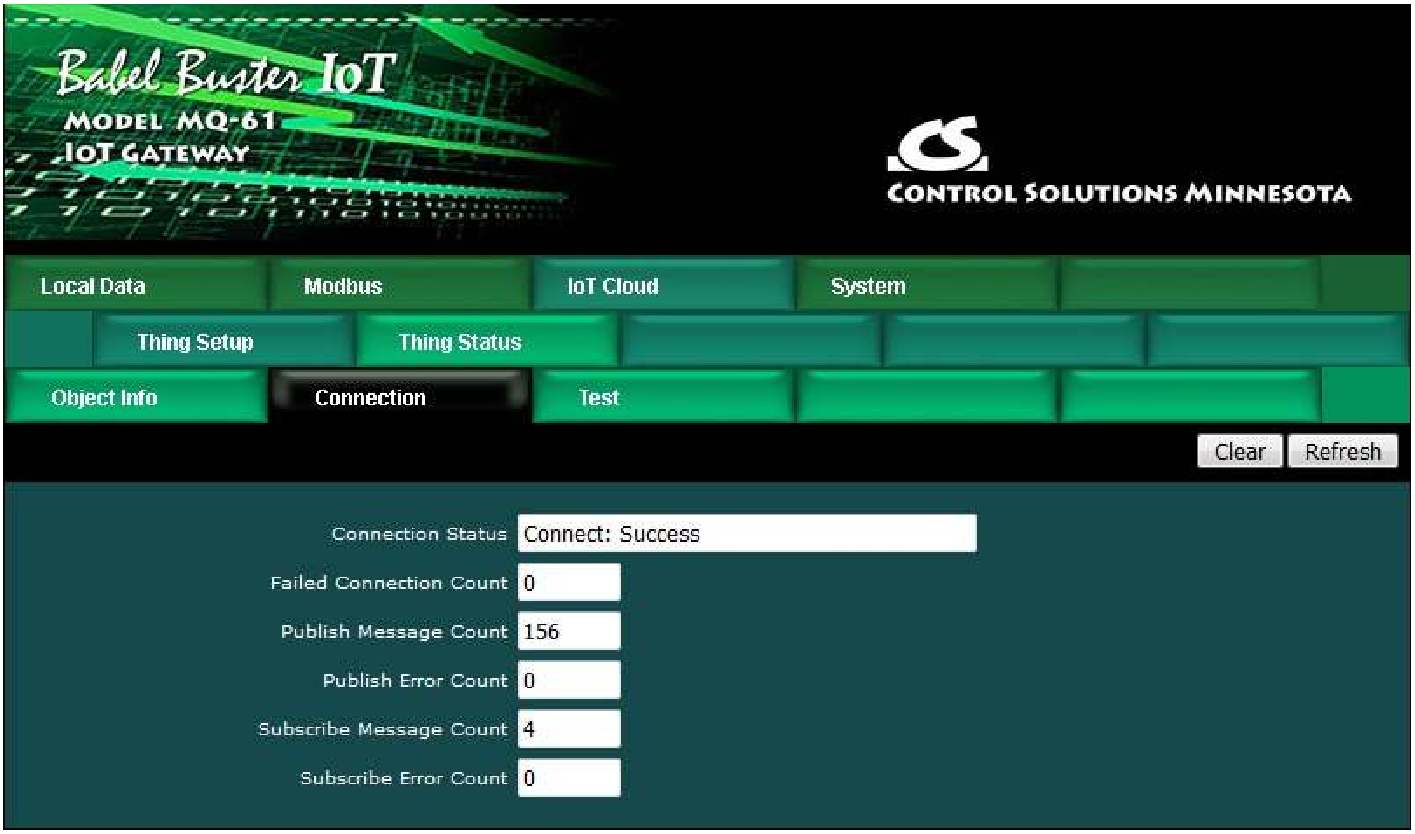
The most recent data exchange for each defined attribute is listed on the Object Info page. The timestamp will show "Pub@" for a published attribute, and "Sub@" when an incoming message was processed for a subscription.



Showing attributes from 1

Atr #	Attribute Name	Last Exchange Timestamp	Last Exchanged Value (or Status)
1	csiSensor1	Pub@ 2020-03-24 17:58:20	70
2	csiSensor2	Pub@ 2020-03-24 17:58:20	10
3	csiSensor3	Pub@ 2020-03-24 17:58:20	48
4	csiSensor4	Pub@ 2020-03-24 17:58:20	55
5	csiSensor5	Pub@ 2020-03-24 17:58:20	40
6	testMessage1	Pub@ 2020-03-24 17:58:29	Test message to cell phone
7			

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



Connection Status: Connect: Success

Failed Connection Count: 0

Publish Message Count: 156

Publish Error Count: 0

Subscribe Message Count: 4

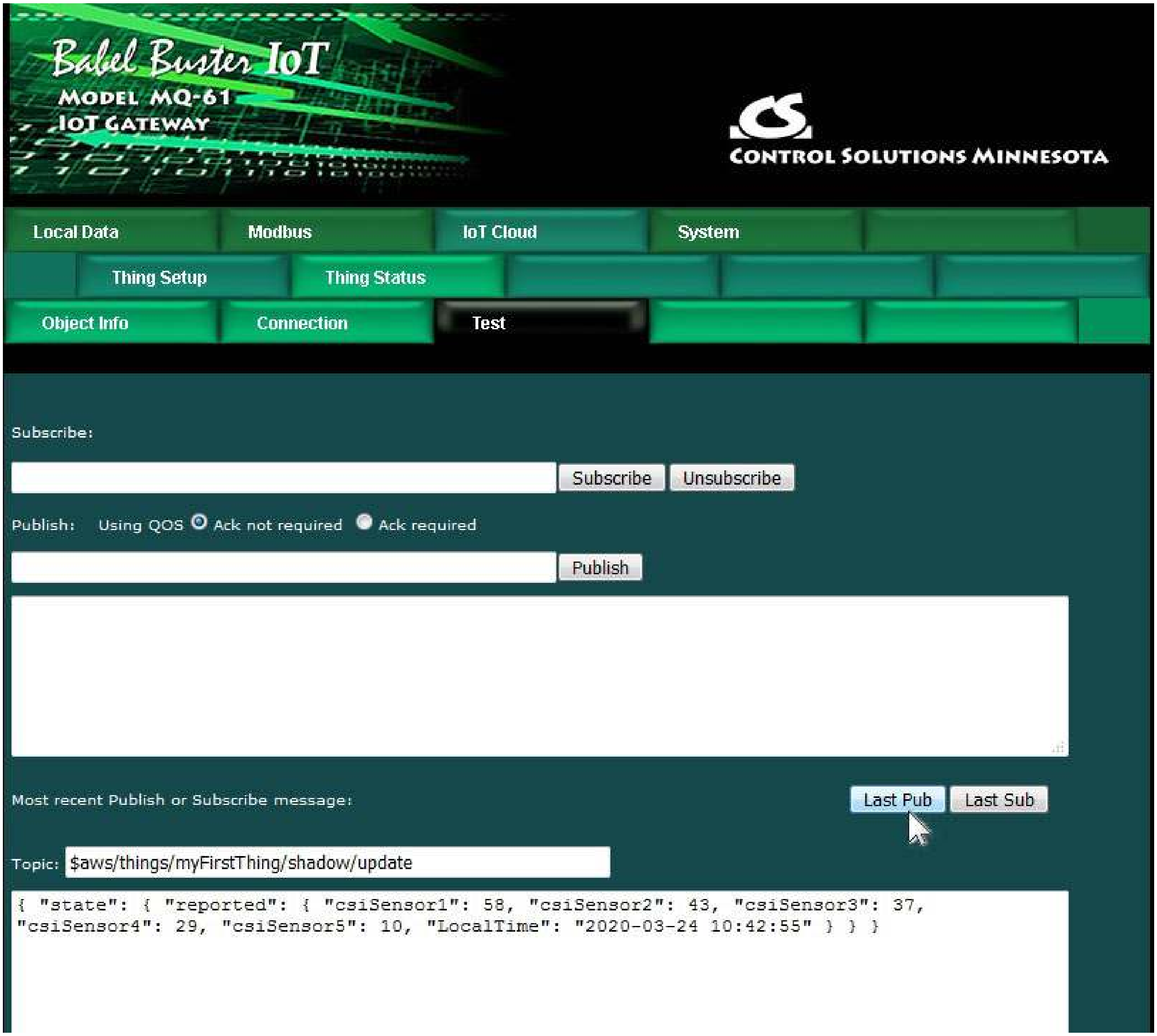
Subscribe Error Count: 0

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 IoT Gateway 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 IoT Gateway, click the Last Pub button. An example of a recent publish message is illustrated below.



Subscribe:

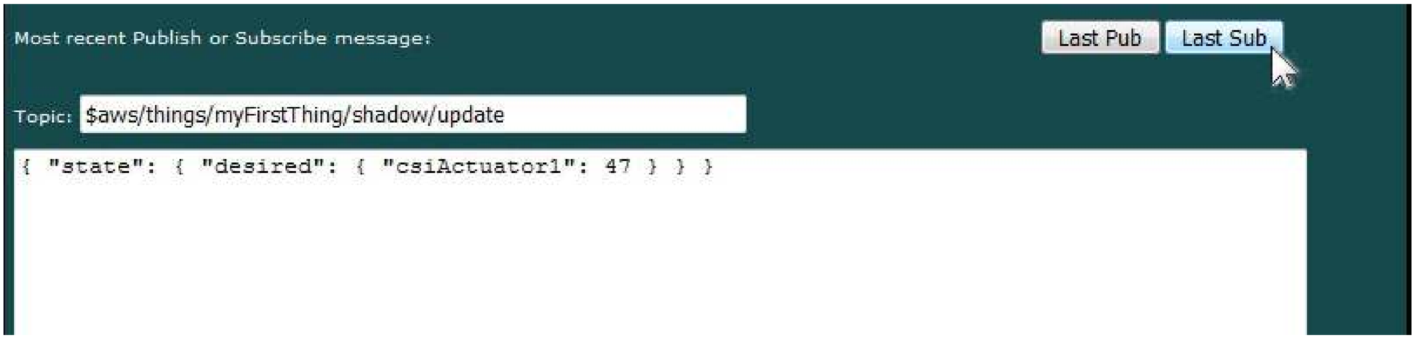
Publish: Using QOS ● Ack not required ○ Ack required

Most recent Publish or Subscribe message:

Topic: $aws/things/myFirstThing/shadow/update

```
{ "state": { "reported": { "csiSensor1": 58, "csiSensor2": 43, "csiSensor3": 37,
"csiSensor4": 29, "csiSensor5": 10, "LocalTime": "2020-03-24 10:42:55" } } }
```

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

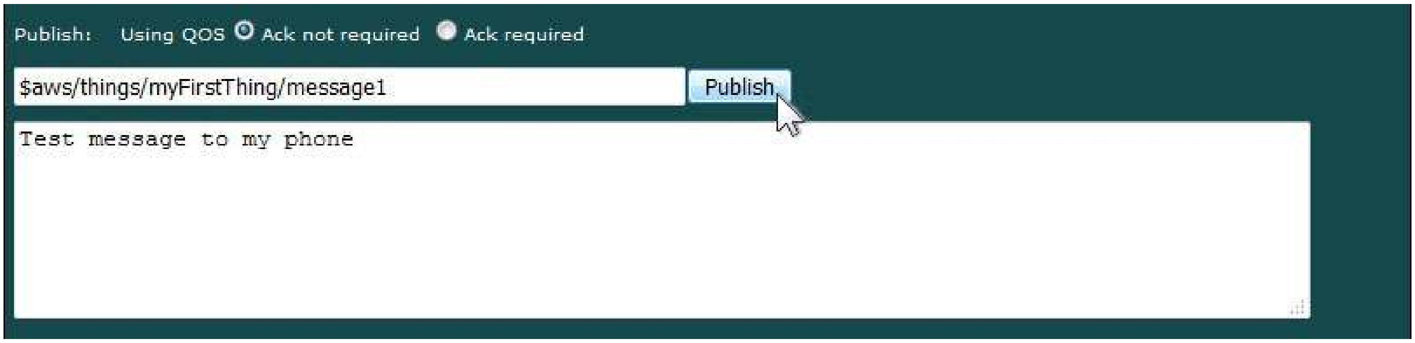


Most recent Publish or Subscribe message:

Topic: $aws/things/myFirstThing/shadow/update

```
{ "state": { "desired": { "csiActuator1": 47 } } }
```

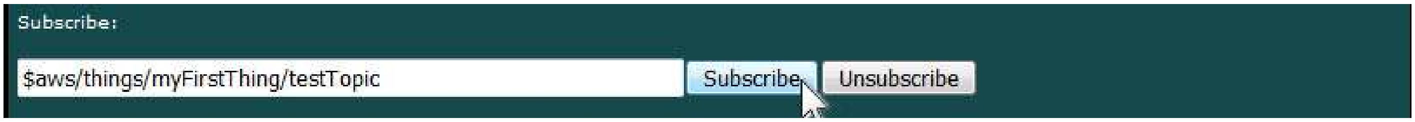
An arbitrary publish message may be sent on this page by entering a topic and payload, and then click the Publish button. This feature should only be used after you have familiarized yourself with MQTT protocol and the AWS side of this connection.



$aws/things/myFirstThing/message1

```
Test message to my phone
```

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



$aws/things/myFirstThing/testTopic

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 Gateway to Publish to AWS

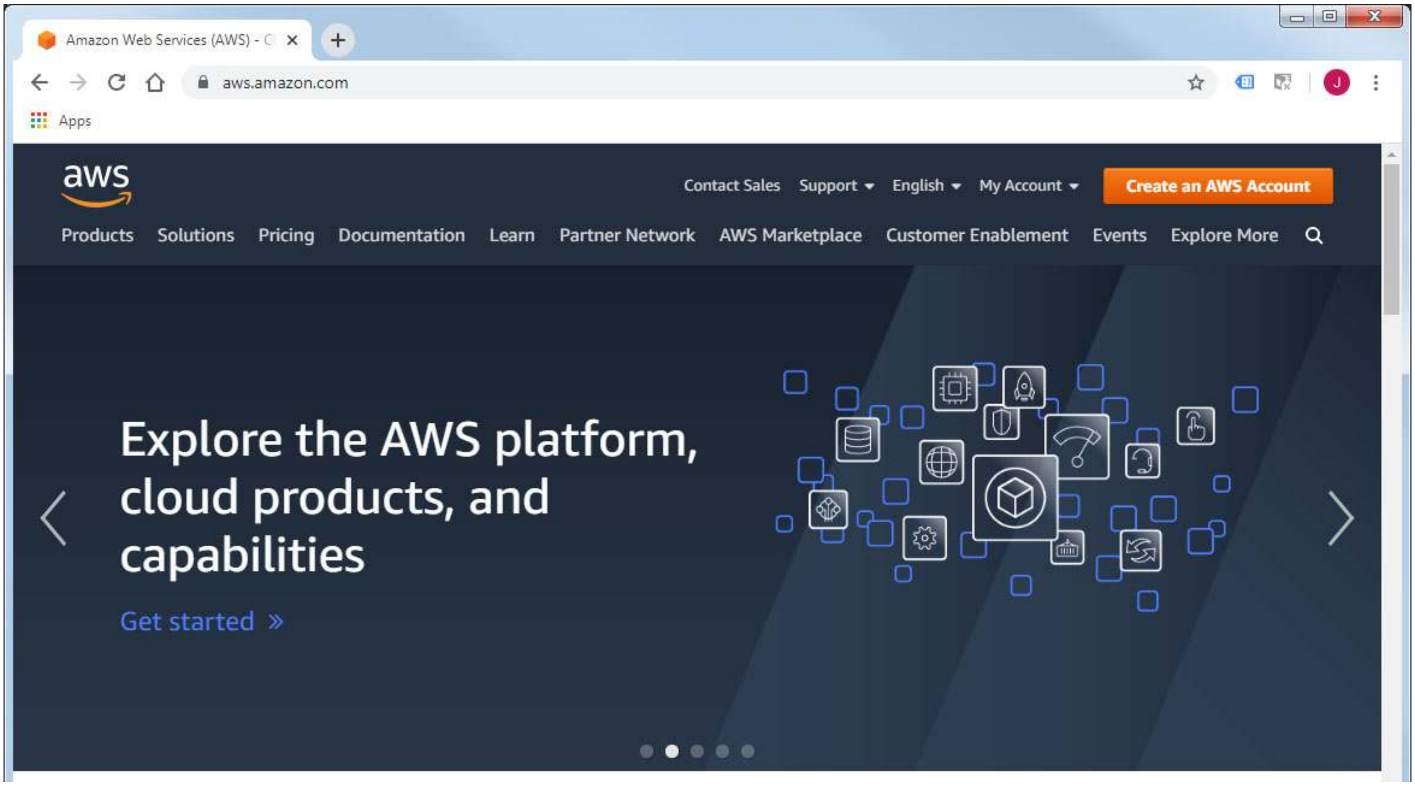
The MQTT term "Publish", from a controls perspective, would be most closely associated with the action of a sensor. You have data available that you wish to transmit to other devices or systems. In the instance we are working with here, we are Publishing data to the AWS server.

Configuring the Babel Buster IoT Gateway to publish data to the Amazon servers requires setting things up on both ends: You need to configure the IoT Gateway, 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.

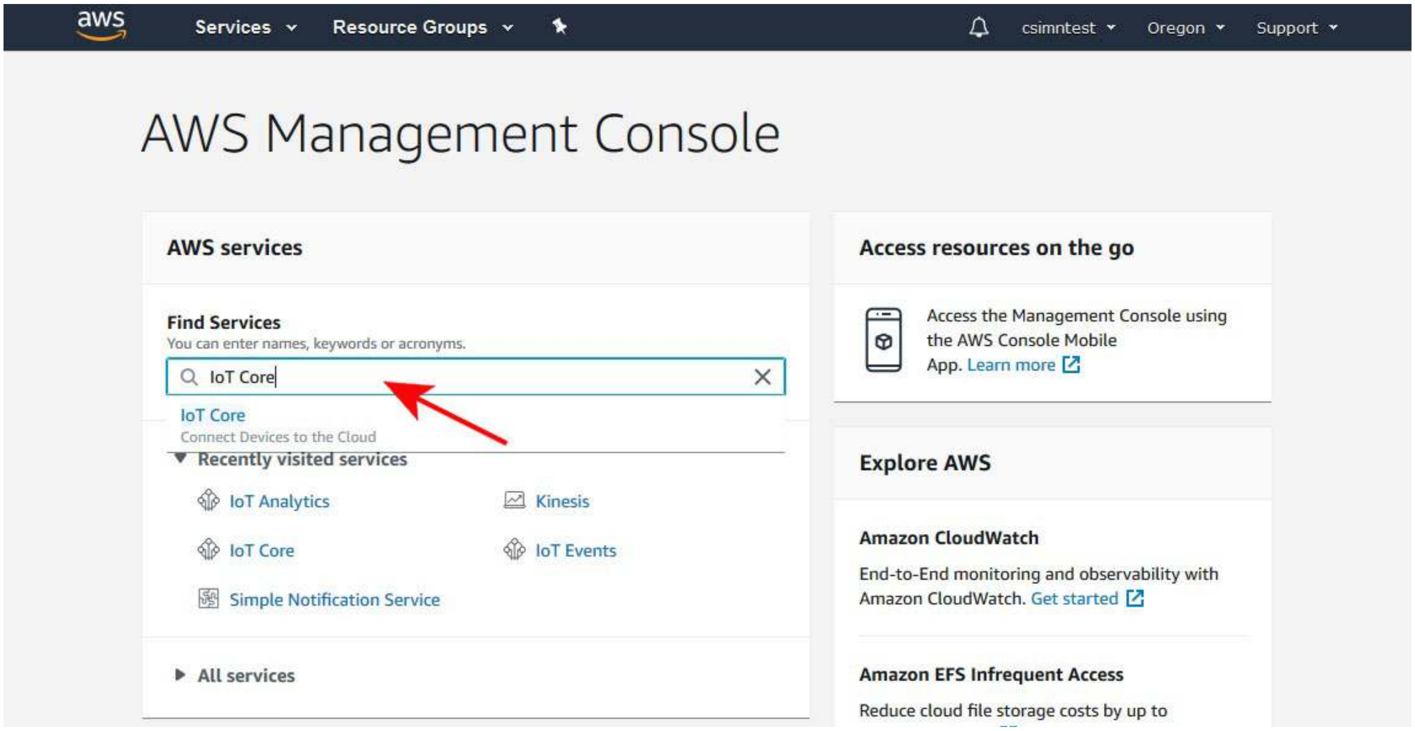
14.1 Create and Register a "Thing"

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.



Explore the AWS platform, cloud products, and capabilities

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



You can enter names, keywords or acronyms.

IoT Core

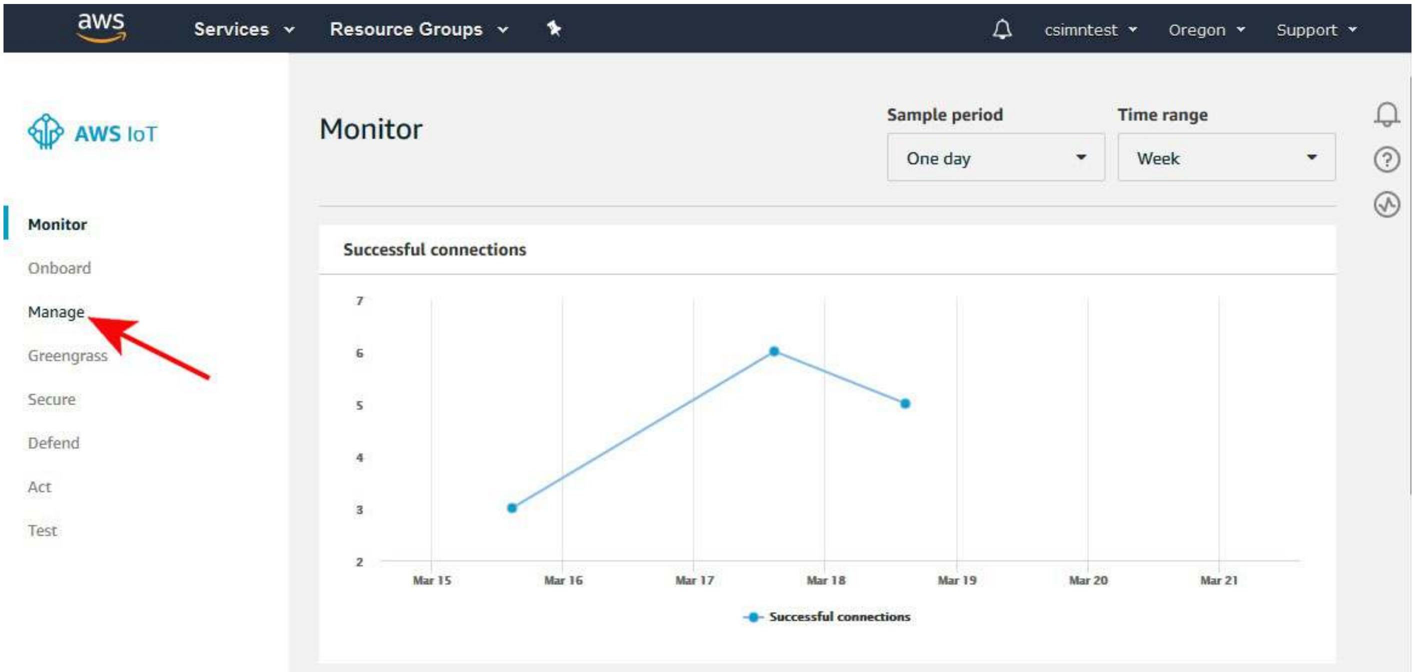
IoT Core
Connect Devices to the Cloud

Access the Management Console using the AWS Console Mobile App. Learn more

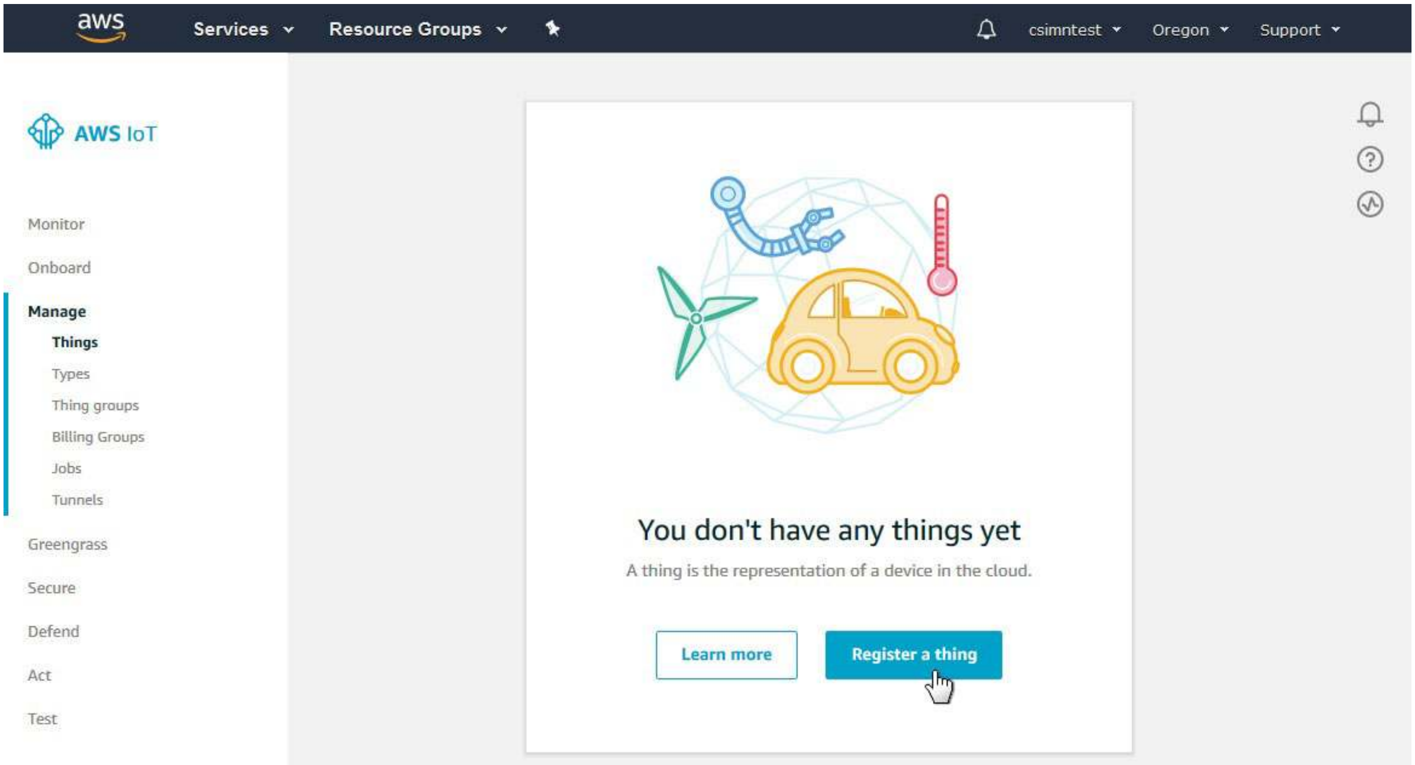
End-to-End monitoring and observability with Amazon CloudWatch. Get started

Reduce cloud file storage costs by up to

From the AWS IoT menu, click on Manage.



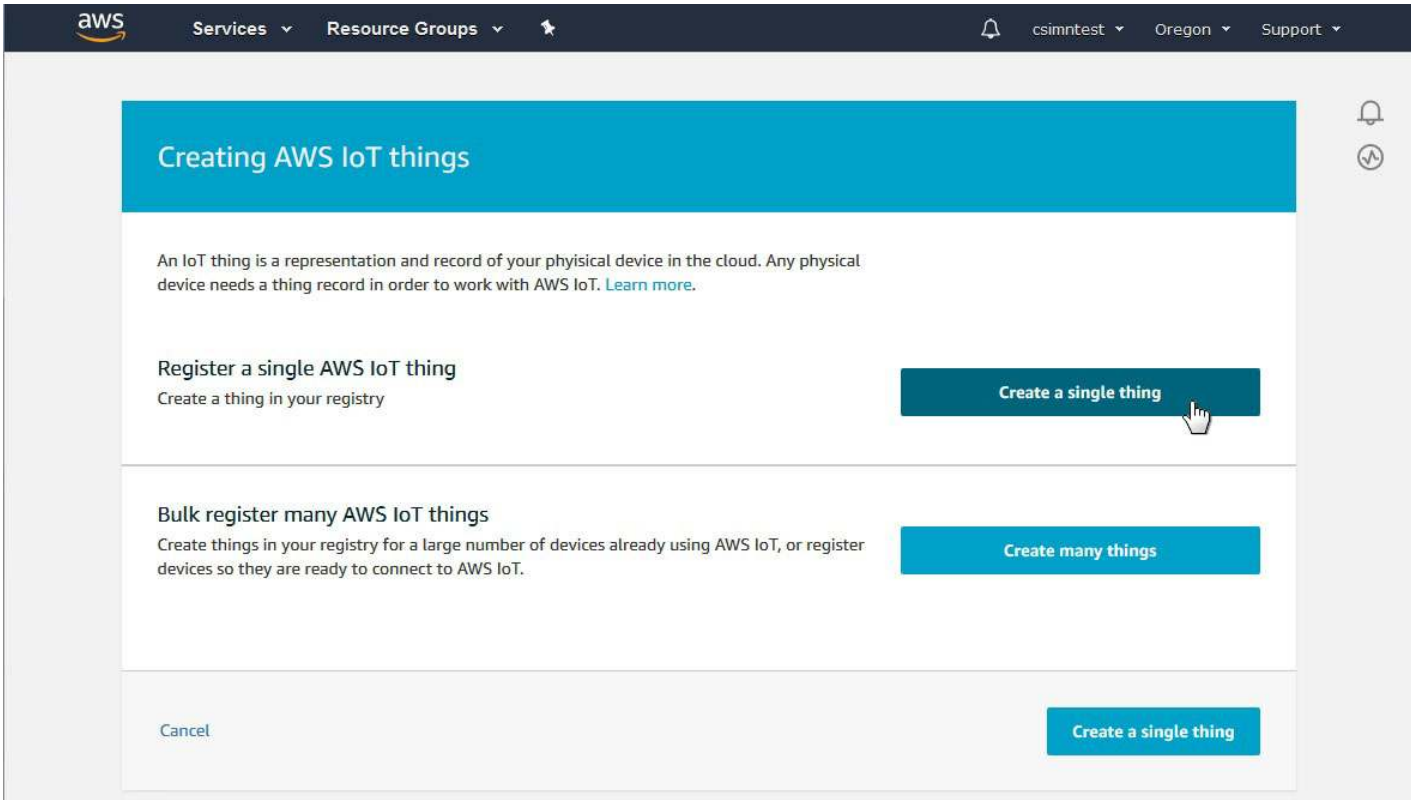
Your initial Things screen will have no Things. Click "Register a thing" to begin the process of creating your first thing.



You don't have any things yet

A thing is the representation of a device in the cloud.

Click "Create a single thing".



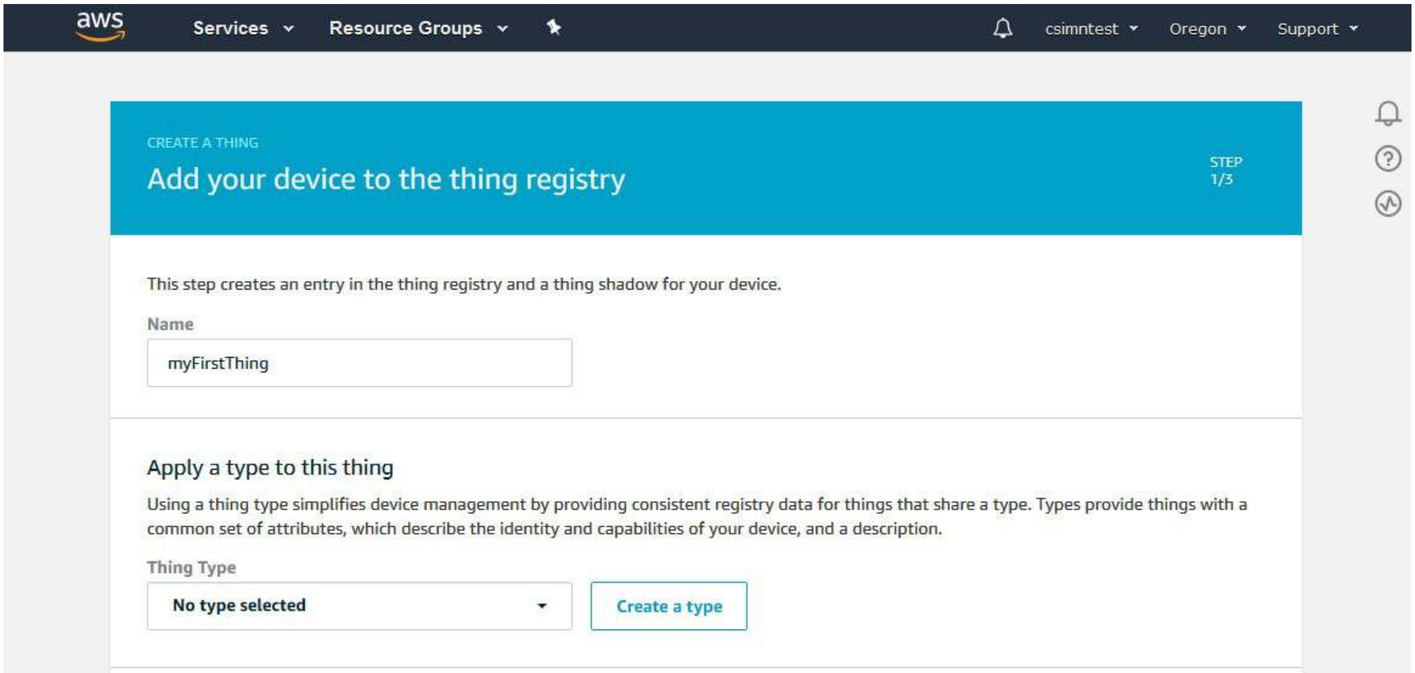
Creating AWS IoT things

An IoT thing is a representation and record of your phyisical device in the cloud. Any physical device needs a thing record in order to work with AWS IoT. Learn more.

Register a single AWS IoT thing
Create a thing in your registry

Bulk register many AWS IoT things
Create things in your registry for a large number of devices already using AWS IoT, or register devices so they are ready to connect to AWS IoT.

Enter a name for your Thing. The name must have only letters and digits, no special characters or spaces. You do not need to create a type at this point. The only entry you need to make here is the thing name.



CREATE A THING

Add your device to the thing registry

STEP 1/3

This step creates an entry in the thing registry and a thing shadow for your device.

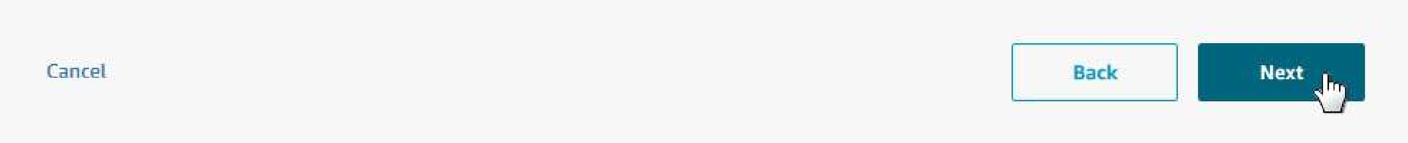
Name: myFirstThing

Apply a type to this thing

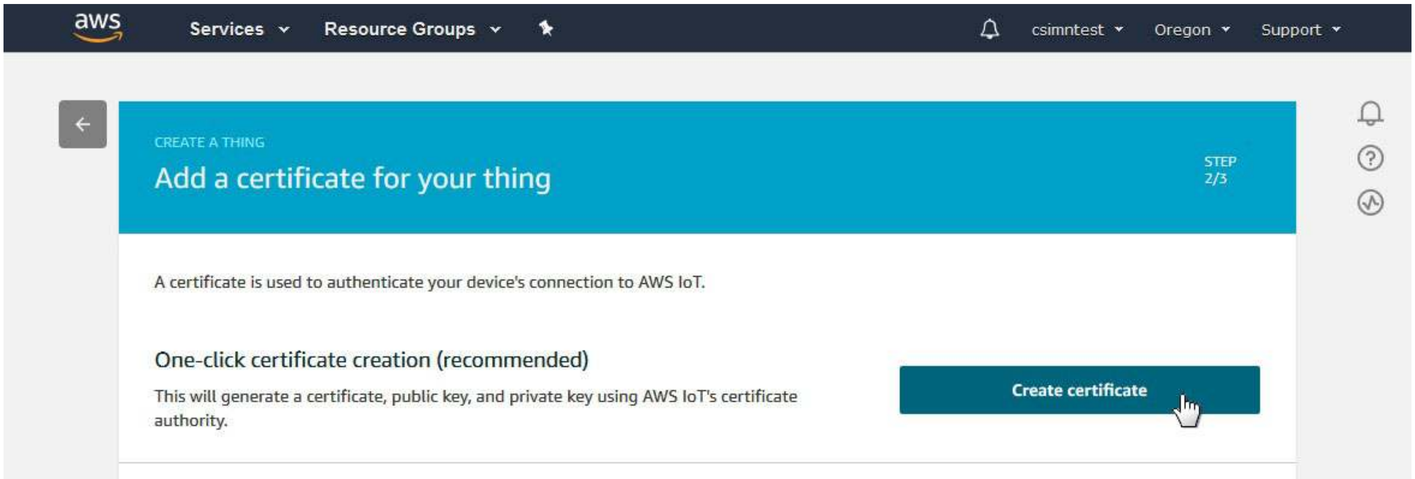
Using a thing type simplifies device management by providing consistent registry data for things that share a type. Types provide things with a common set of attributes, which describe the identity and capabilities of your device, and a description.

Thing Type: No type selected

Click on Next to continue the process.



The connection between the IoT Gateway and the AWS server is a secure connection requiring SSL certificates. These will be created for you by the AWS system. Click on "Create certificate".



CREATE A THING

Add a certificate for your thing

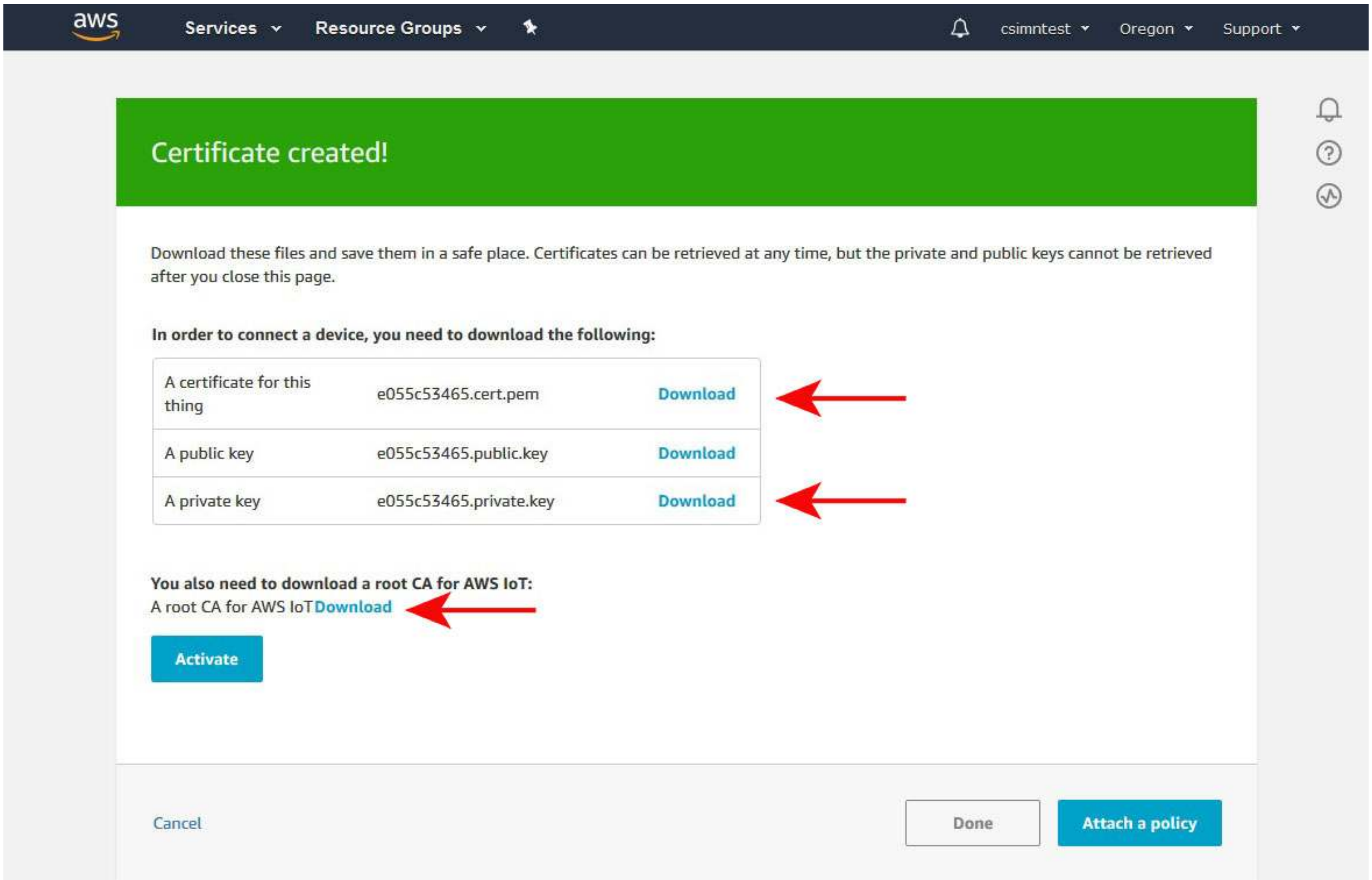
A certificate is used to authenticate your device's connection to AWS IoT.

One-click certificate creation (recommended)

This will generate a certificate, public key, and private key using AWS IoT's certificate authority.

Create certificate

Upon creating certificates, you now need to download them. Download the certificate for this thing, and the private key. You will also need the root CA for your certificates. Click the Download link for the root CA to follow that path.



Certificate created!

Download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys cannot be retrieved after you close this page.

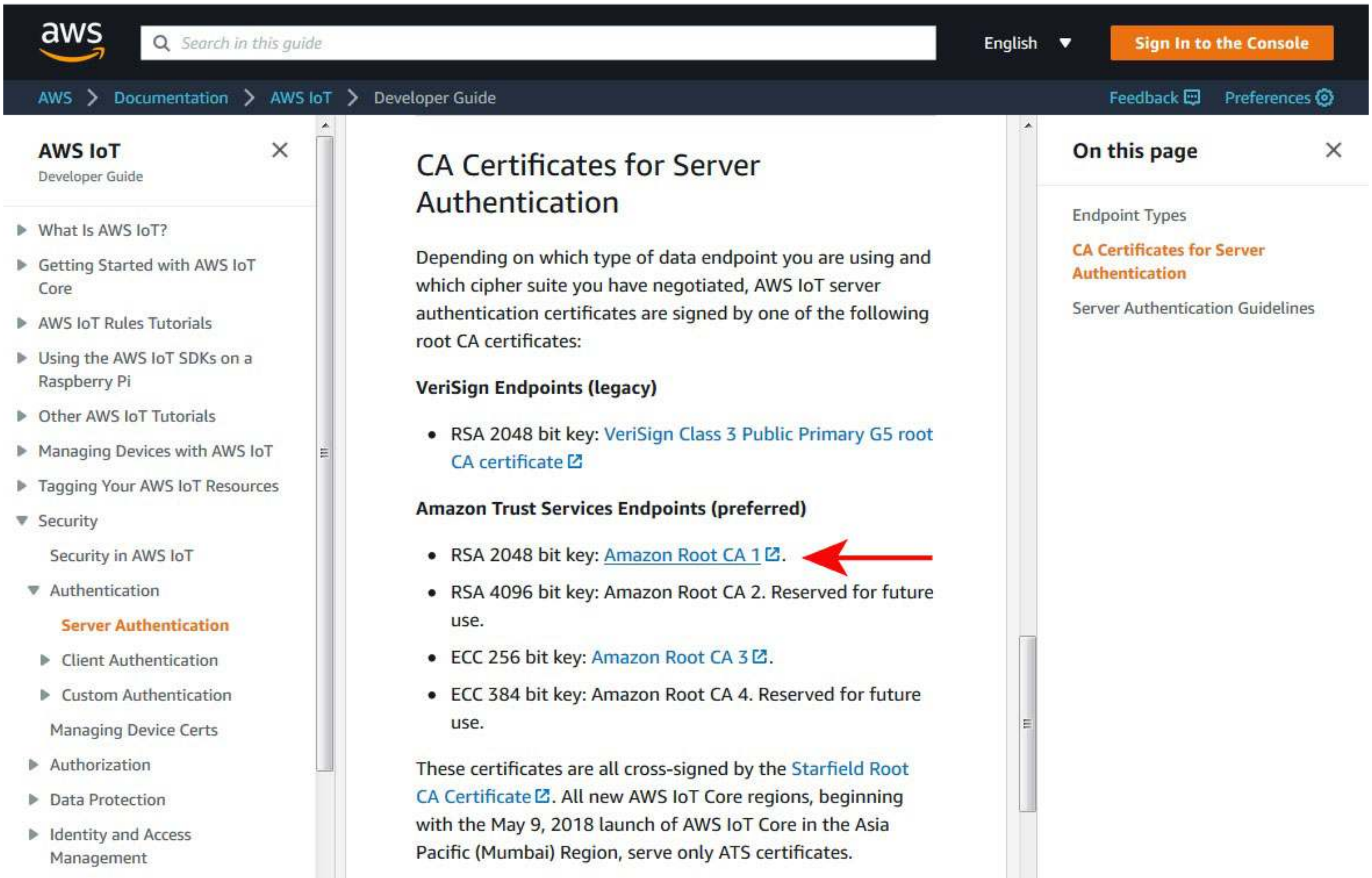
In order to connect a device, you need to download the following:

A certificate for this thing	e055c53465.cert.pem	Download
A public key	e055c53465.public.key	Download
A private key	e055c53465.private.key	Download

You also need to download a root CA for AWS IoT:

A root CA for AWS IoT Download

The root CA download link will take you to the CA certificates page. Click on the link for Amazon Root CA 1.



CA Certificates for Server Authentication

Depending on which type of data endpoint you are using and which cipher suite you have negotiated, AWS IoT server authentication certificates are signed by one of the following root CA certificates:

VeriSign Endpoints (legacy)

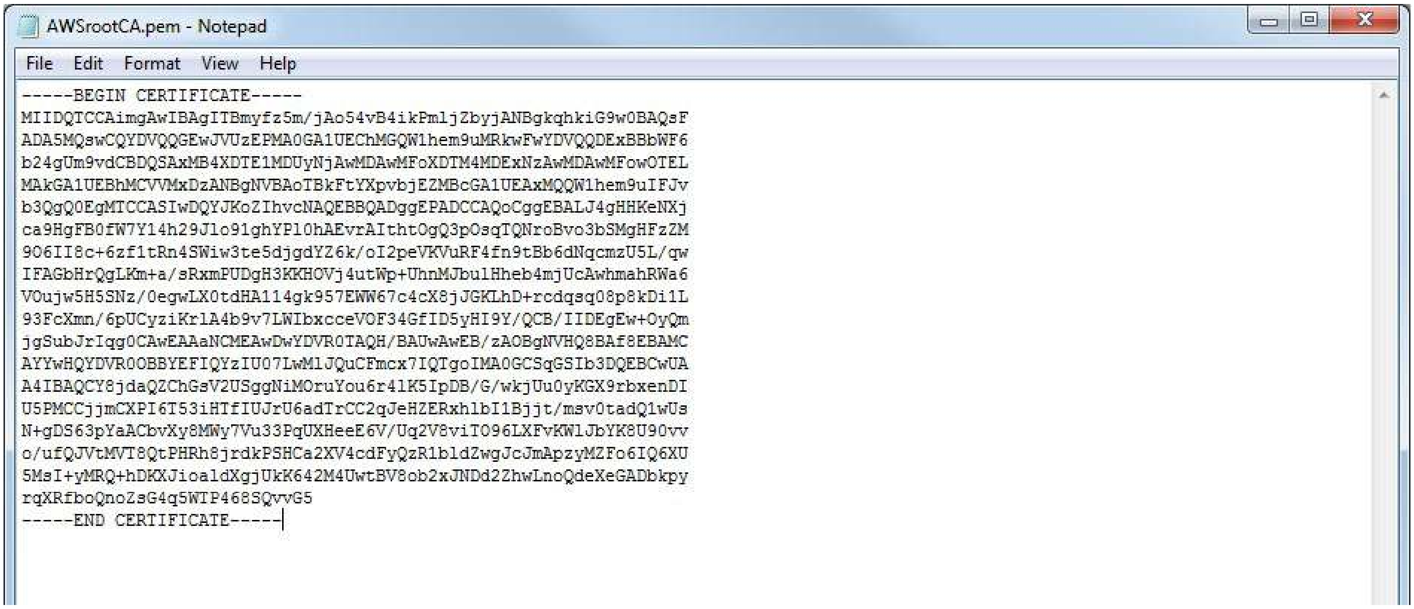
- RSA 2048 bit key: VeriSign Class 3 Public Primary G5 root CA certificate

Amazon Trust Services Endpoints (preferred)

- RSA 2048 bit key: Amazon Root CA 1.
- RSA 4096 bit key: Amazon Root CA 2. Reserved for future use.
- ECC 256 bit key: Amazon Root CA 3.
- ECC 384 bit key: Amazon Root CA 4. Reserved for future use.

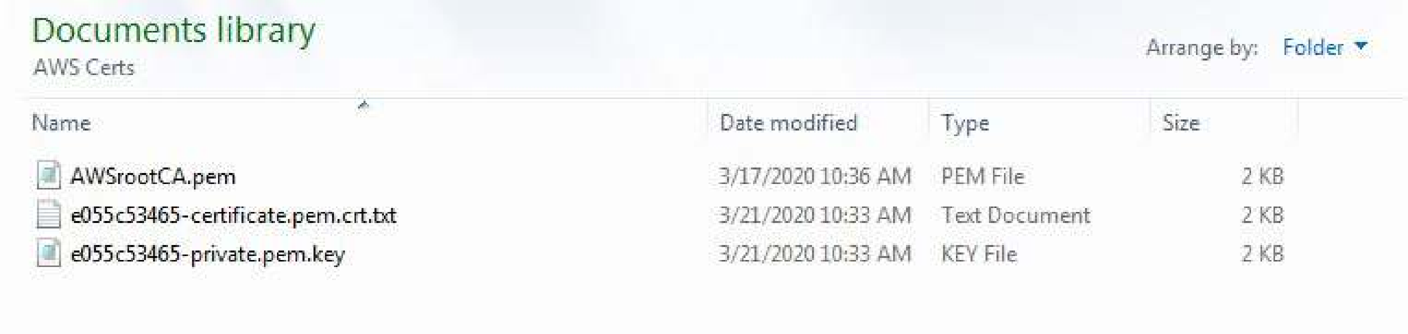
These certificates are all cross-signed by the Starfield Root CA Certificate. All new AWS IoT Core regions, beginning with the May 9, 2018 launch of AWS IoT Core in the Asia Pacific (Mumbai) Region, serve only ATS certificates.

Clicking the Amazon Root CA 1 link will open a page with the certificate in text form. Copy this certificate and paste into a blank Notepad document (or other plain text editor). Then save this as "AWSrootCA.pem" or similar name - just keep track of the fact that this is your root CA certificate.



```
-----BEGIN CERTIFICATE-----
MIIDQTCCAimgAwIBAgITBmyfz5m/jAo54vB4ikPmljZbyjANBgkqhkiG9w0BAQsF
ADA5MQswCQYDVQQGEwJVUzEPMA0GA1UEChMGQW1hem9uMRkwFwYDVQQDExBBbWF6
b24gUm9vdCBDQSAxMB4XDTE1MDUyNjAwMDAwMFoXDTM4MDExNzAwMDAwMFowOTEL
MAkGA1UEBhMCVVMxDzANBgNVBAoTBkFtYXpvbjEZMBcGA1UEAxMQQW1hem9uIFJv
b3QgQ0EgMTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBALJ4gHHKeNXj
ca9HgFB0fW7Y14h29Jlo91ghYPl0hAEvrAIthtOgQ3pOsqTQNroBvo3bSMgHFzZM
9O6II8c+6zf1tRn4SWiw3te5djgdYZ6k/oI2peVKVuRF4fn9tBb6dNqcmzU5L/qw
IFAGbHrQgLKm+a/sRxmPUDgH3KKHOVj4utWp+UhnMJbulHheb4mjUcAwhmahRWa6
VOujw5H5SNz/0egwLX0tdHA114gk957EWW67c4cX8jJGKLhD+rcdqsq08p8kDi1L
93FcXmn/6pUCyziKrlA4b9v7LWIbxcceVOF34GfID5yHI9Y/QCB/IIDEgEw+OyQm
jgSubJrIqg0CAwEAAaNCMEAwDwYDVR0TAQH/BAUwAwEB/zAOBgNVHQ8BAf8EBAMC
AYYwHQYDVR0OBBYEFIQYzIU07LwMlJQuCFmcx7IQTgoIMA0GCSqGSIb3DQEBCwUA
A4IBAQCY8jdaQZChGsV2USggNiMOruYou6r4lK5IpDB/G/wkjUu0yKGX9rbxenDI
U5PMCCjjmCXPI6T53iHTfIUJrU6adTrCC2qJeHZERxhlbI1Bjjt/msv0tadQ1wUs
N+gDS63pYaACbvXy8MWy7Vu33PqUXHeeE6V/Uq2V8viTO96LXFvKWlJbYK8U90vv
o/ufQJVtMVT8QtPHRh8jrdkPSHCa2XV4cdFyQzR1bldZwgJcJmApzyMZFo6IQ6XU
5MsI+yMRQ+hDKXJioaldXgjUkK642M4UwtBV8ob2xJNDd2ZhwLnoQdeXeGADbkpy
rqXRfboQnoZsG4q5WTP468SQvvG5
-----END CERTIFICATE-----
```

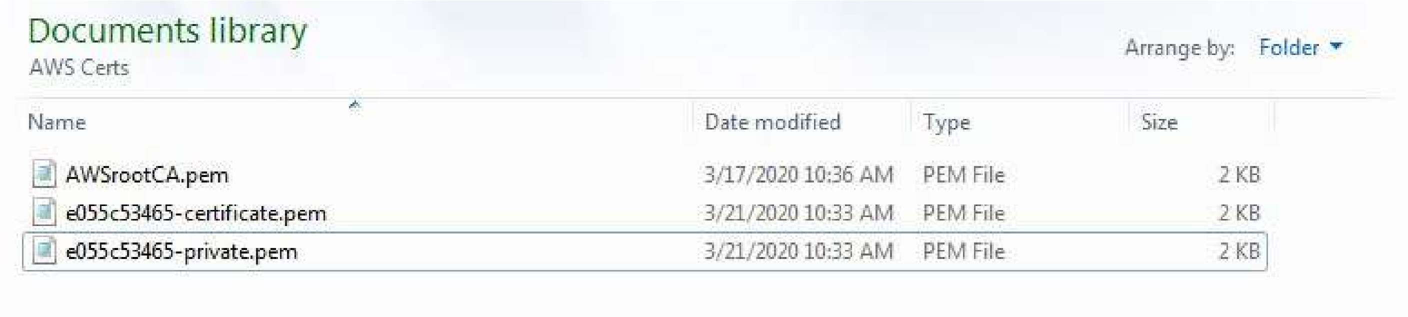
Collect up the three SSL certificates you have now obtained, place them somewhere you can find back easily when uploading them to the IoT Gateway.



Documents library
AWS Certs

Name	Date modified	Type	Size
AWSrootCA.pem	3/17/2020 10:36 AM	PEM File	2 KB
e055c53465-certificate.pem.crt.txt	3/21/2020 10:33 AM	Text Document	2 KB
e055c53465-private.pem.key	3/21/2020 10:33 AM	KEY File	2 KB

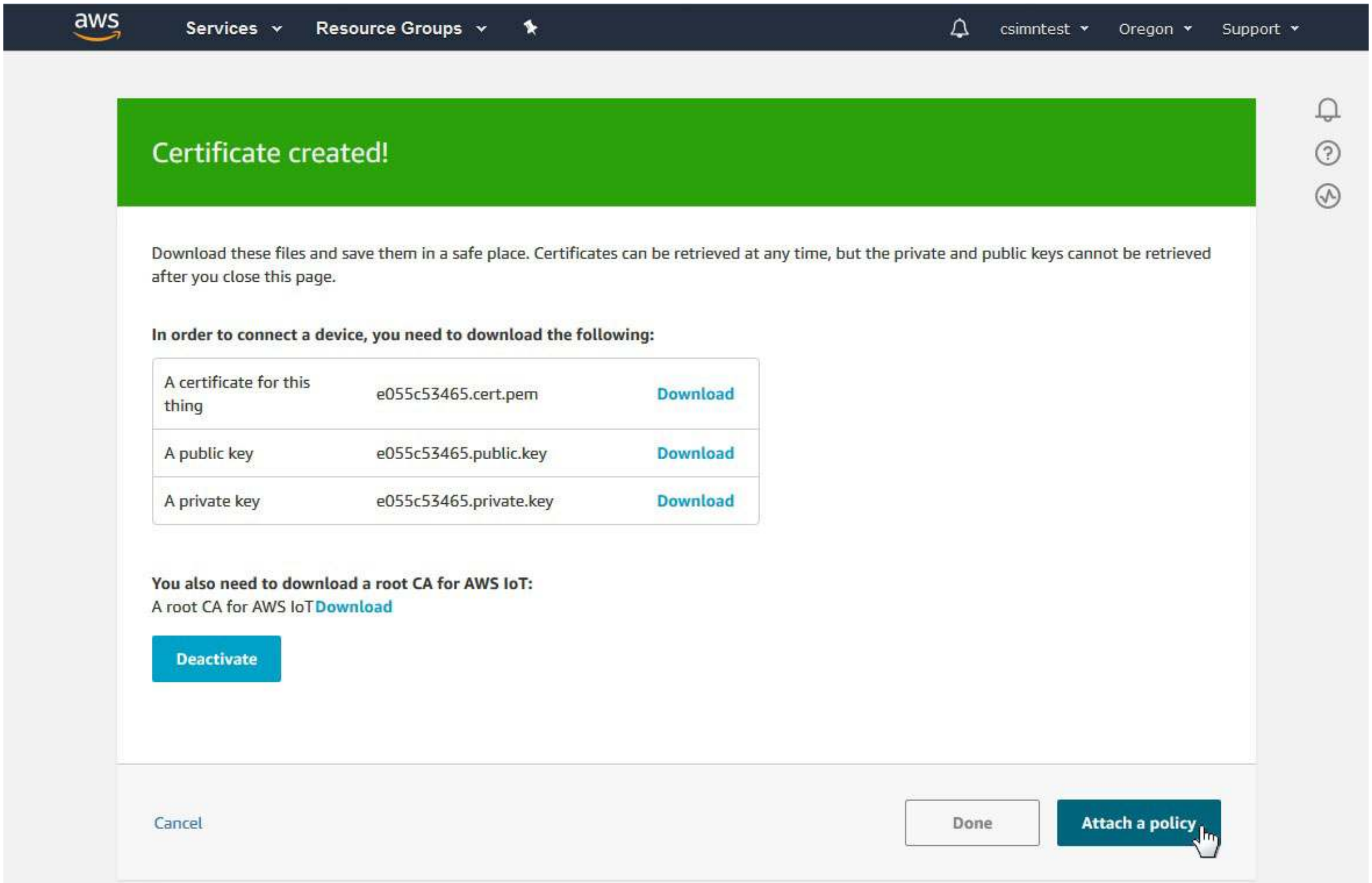
Make minor renaming changes to these files such that their suffix is ".pem" in all cases.



Documents library
AWS Certs

Name	Date modified	Type	Size
AWSrootCA.pem	3/17/2020 10:36 AM	PEM File	2 KB
e055c53465-certificate.pem	3/21/2020 10:33 AM	PEM File	2 KB
e055c53465-private.pem	3/21/2020 10:33 AM	PEM File	2 KB

Either before or after downloading the certificates, click the Activate button to activate those SSL certificates on the AWS server. (This does not change the content of the certificates.) Click on Attach a Policy to move on.



Certificate created!

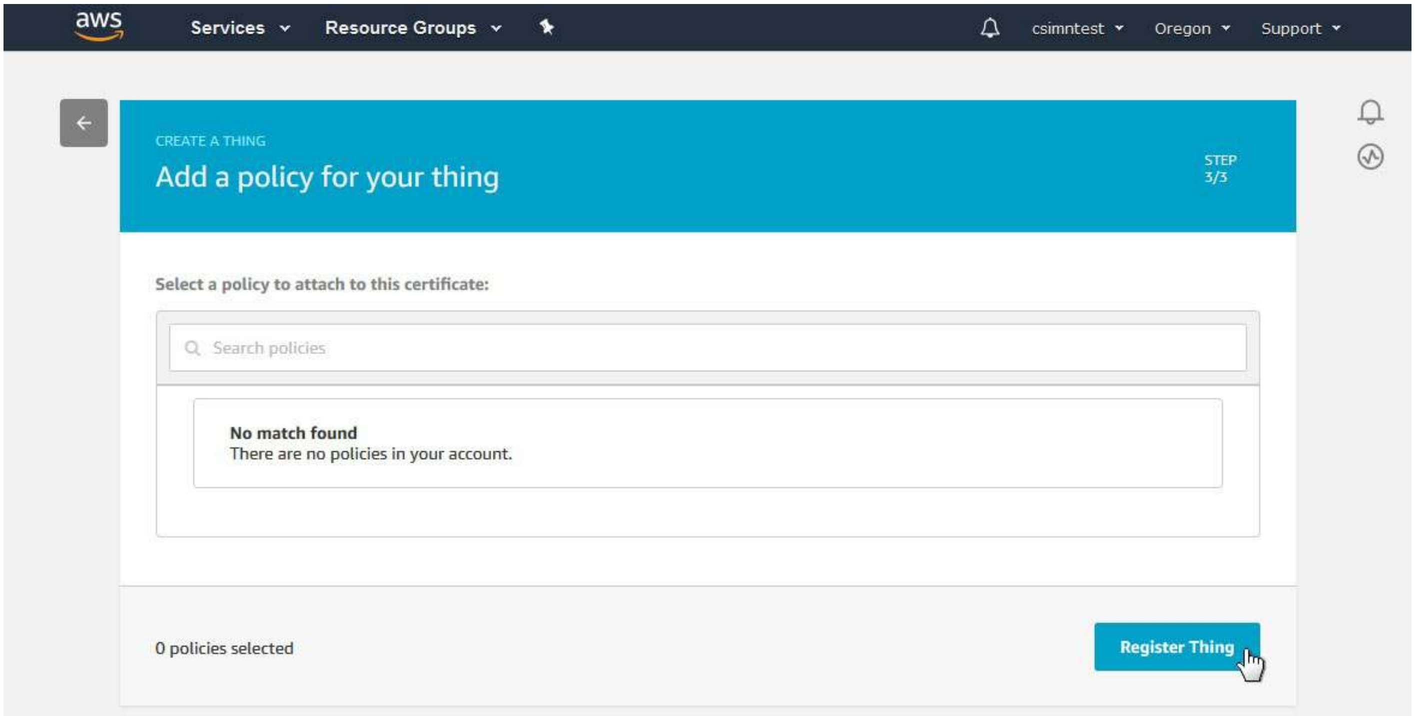
Download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys cannot be retrieved after you close this page.

In order to connect a device, you need to download the following:

A certificate for this thing	e055c53465.cert.pem	Download
A public key	e055c53465.public.key	Download
A private key	e055c53465.private.key	Download

You also need to download a root CA for AWS IoT:
A root CA for AWS IoT Download

You do not yet have any policies to attach, so click Register Thing.



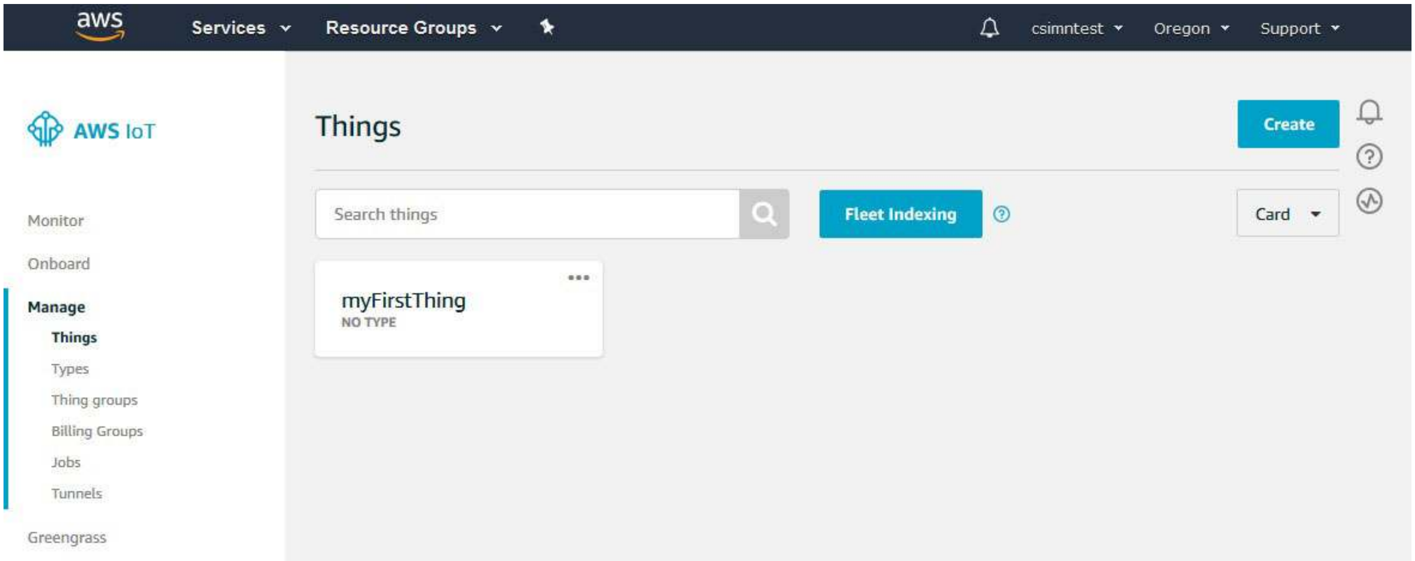
Add a policy for your thing

Select a policy to attach to this certificate:

No match found
There are no policies in your account.

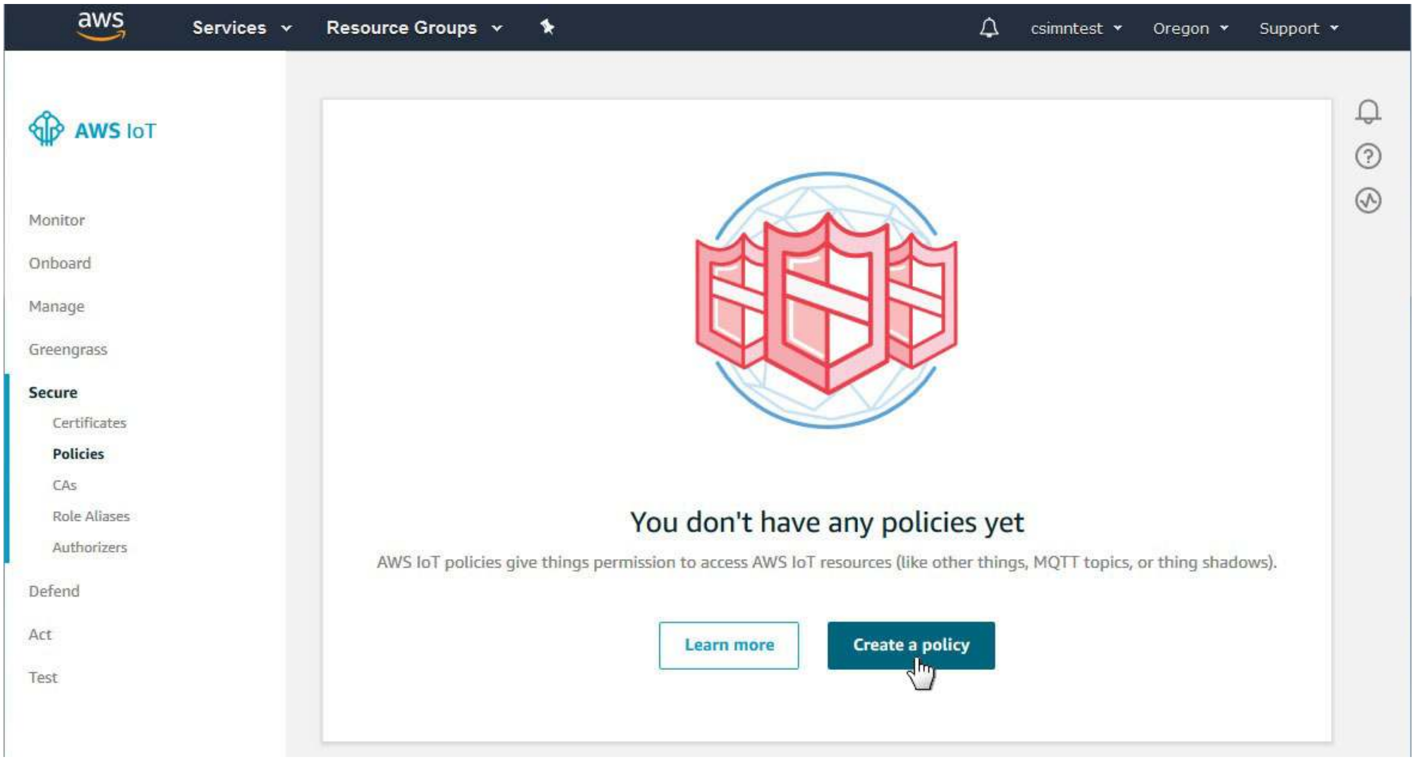
0 policies selected

Your Things page should now look like this.



14.2 Create a Policy

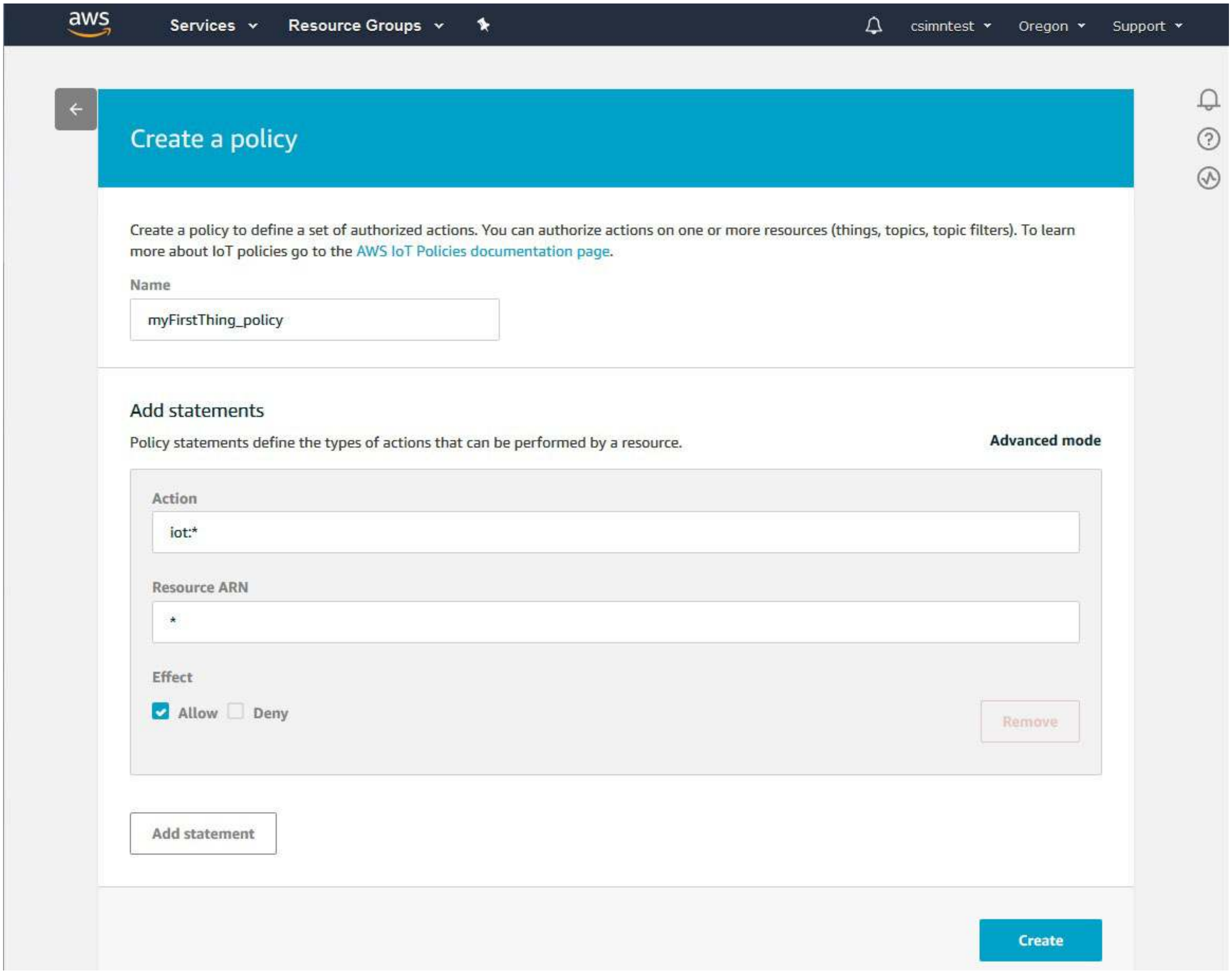
The Thing requires a Policy before the AWS server will respond to communication originated by the IoT Gateway. You find Policies under the Secure category on the AWS IoT menu. Click on Create a policy.



You don't have any policies yet

AWS IoT policies give things permission to access AWS IoT resources (like other things, MQTT topics, or thing shadows).

Enter a name for your policy. The minimum statement you need to provide is an Action of "iot:*" and Resource ARN of "*" (which means all). Effect should be Allow. Make these entries and selections and then click Create.



Create a policy

Create a policy to define a set of authorized actions. You can authorize actions on one or more resources (things, topics, topic filters). To learn more about IoT policies go to the AWS IoT Policies documentation page.

Name: myFirstThing_policy

Add statements

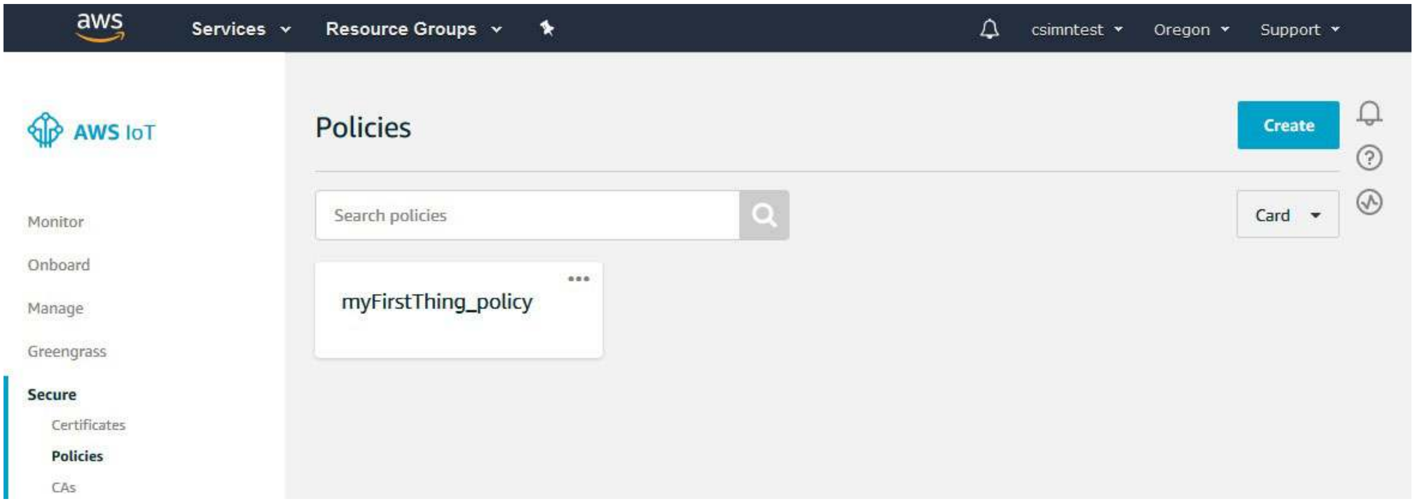
Policy statements define the types of actions that can be performed by a resource.

Action: iot:*

Resource ARN: *

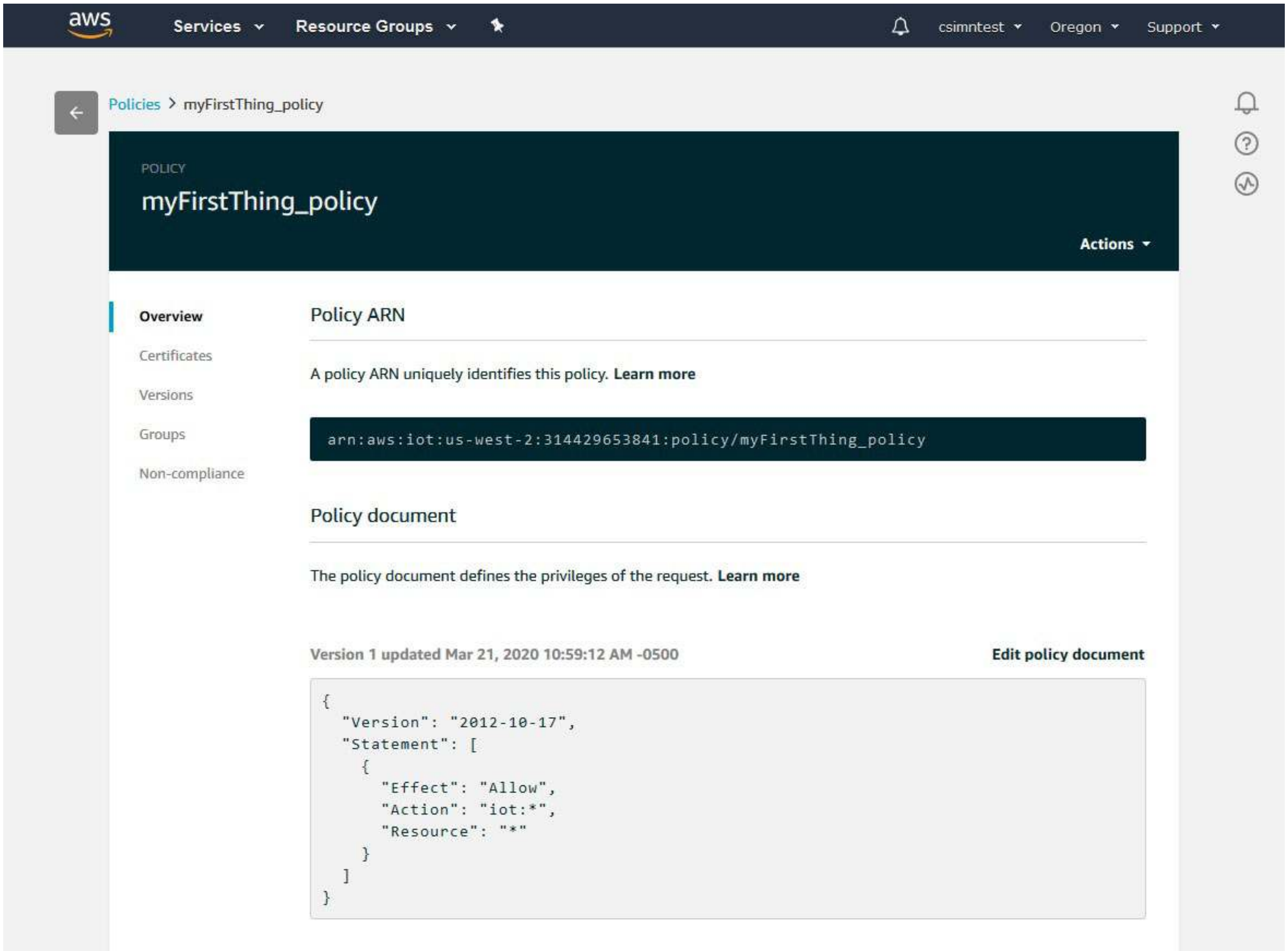
Effect: ☑ Allow □ Deny

Your policies page will now look like this.



myFirstThing_policy

If you click on your policy, you can review what it now looks like.



myFirstThing_policy

Policy ARN

A policy ARN uniquely identifies this policy. Learn more

```
arn:aws:iot:us-west-2:314429653841:policy/myFirstThing_policy
```

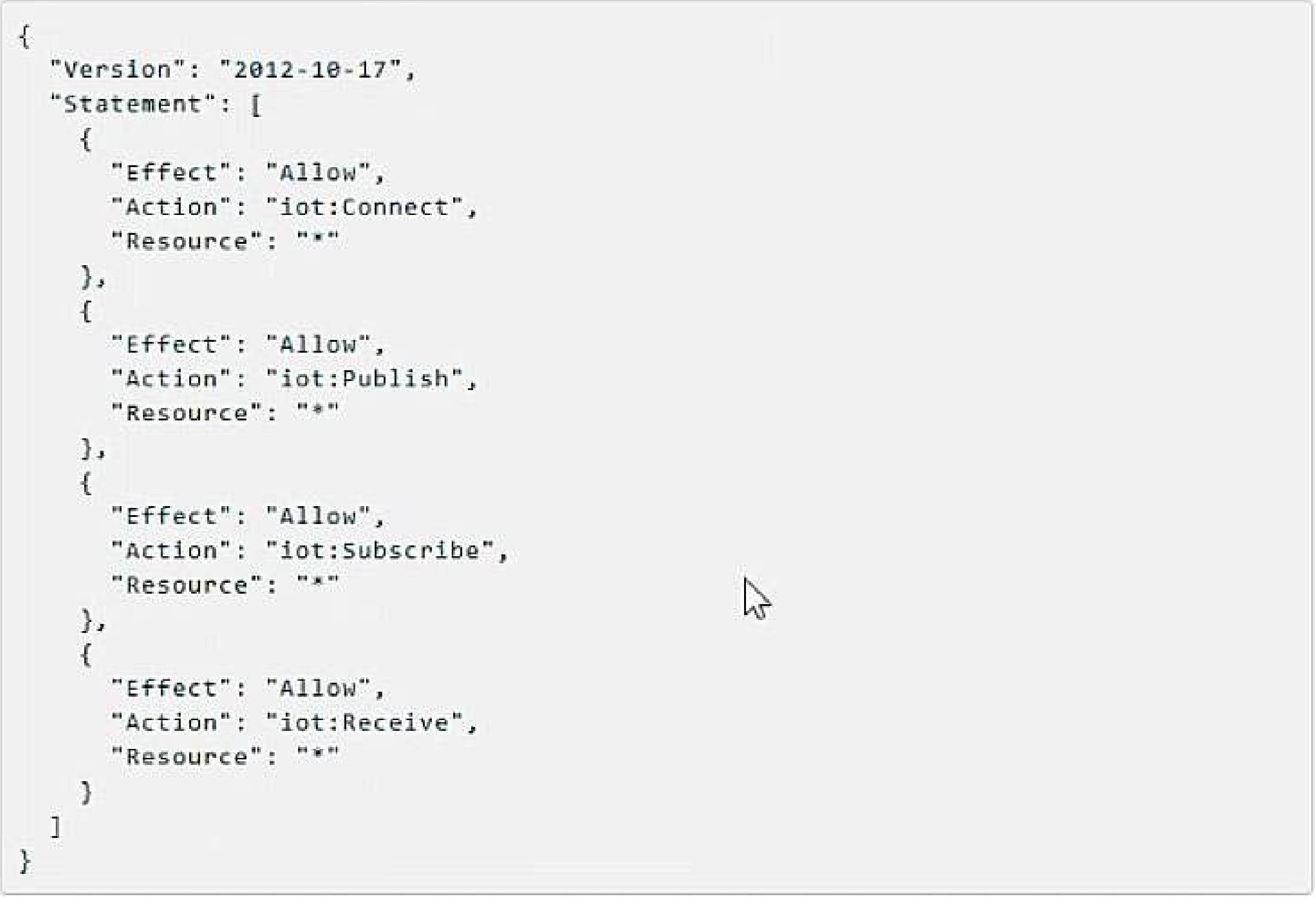
Policy document

The policy document defines the privileges of the request. Learn more

Version 1 updated Mar 21, 2020 10:59:12 AM -0500

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

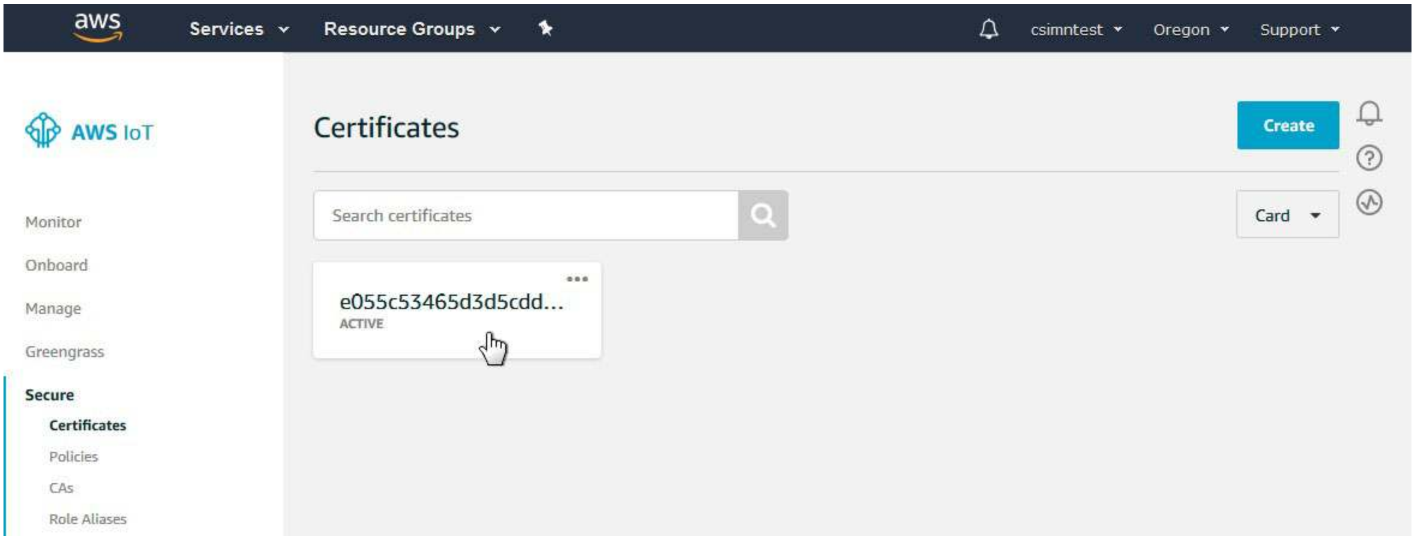
The most unrestrictive policy is illustrated above. A more complete policy also often used as a default is illustrated below.



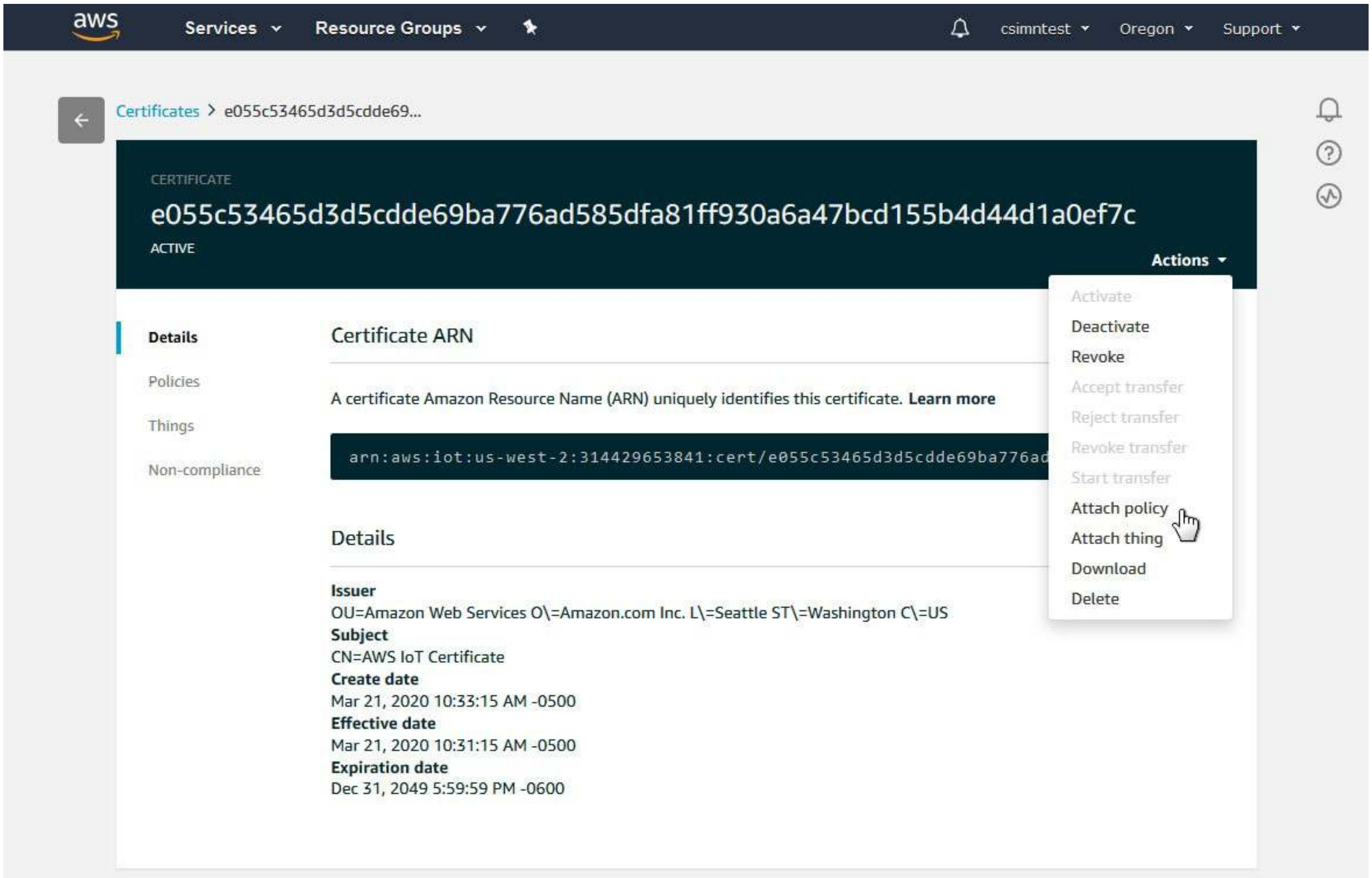
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Publish",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
    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

14.3 Attach Policy and Thing to SSL Certificates

Now that you have a policy, it is necessary to attach that policy and your thing to the SSL certificates you created. Go to the Certificates page and click on your certificate.



The certificate page will appear as illustrated below. Click on Attach policy.



CERTIFICATE

e055c53465d3d5cdde69ba776ad585dfa81ff930a6a47bcd155b4d44d1a0ef7c

ACTIVE

Certificate ARN

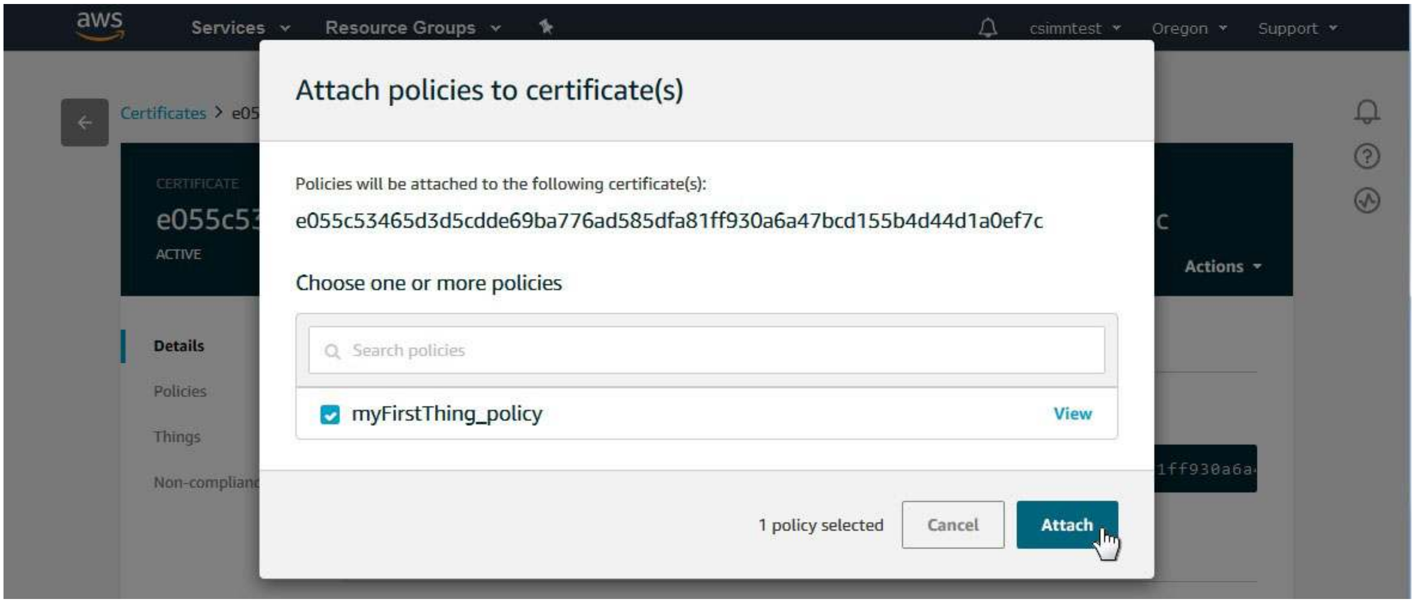
A certificate Amazon Resource Name (ARN) uniquely identifies this certificate. **Learn more**

```
arn:aws:iot:us-west-2:314429653841:cert/e055c53465d3d5cdde69ba776ad
```

Details

Issuer
OU=Amazon Web Services O\=Amazon.com Inc. L\=Seattle ST\=Washington C\=US
Subject
CN=AWS IoT Certificate
Create date
Mar 21, 2020 10:33:15 AM -0500
Effective date
Mar 21, 2020 10:31:15 AM -0500
Expiration date
Dec 31, 2049 5:59:59 PM -0600

Clicking on "Attach policy" will bring up the window illustrated below. Select your policy and click Attach.



Attach policies to certificate(s)

Policies will be attached to the following certificate(s):

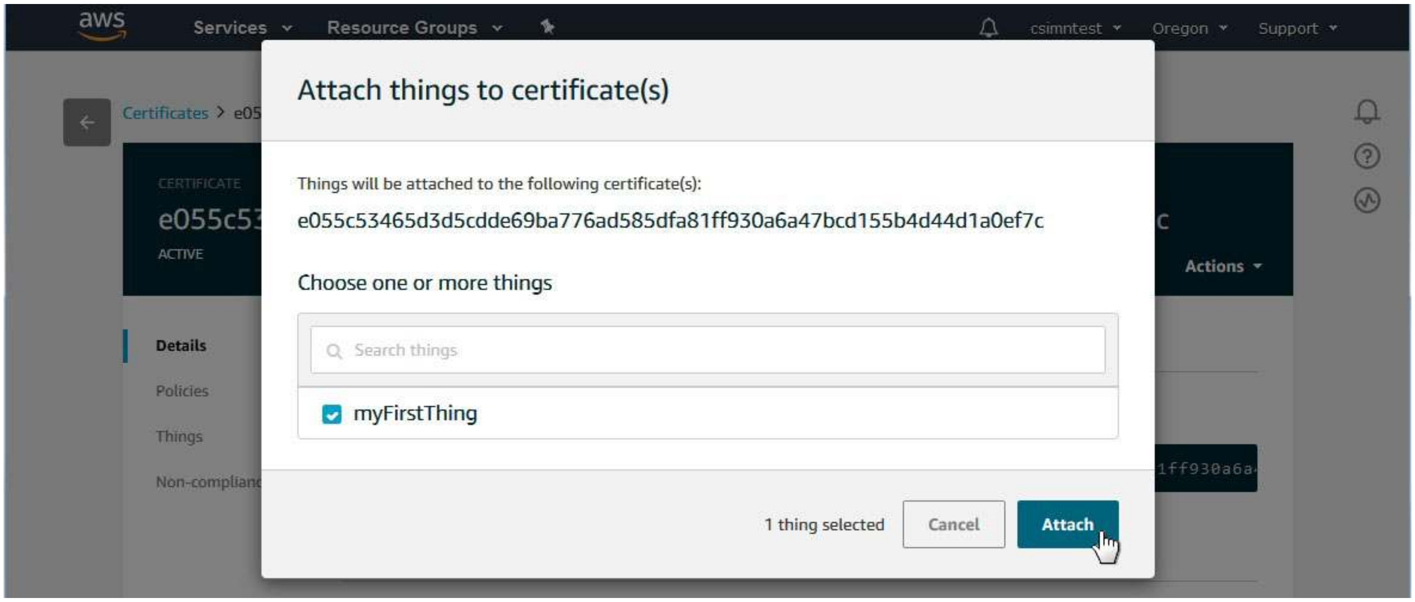
e055c53465d3d5cdde69ba776ad585dfa81ff930a6a47bcd155b4d44d1a0ef7c

Choose one or more policies

☑ myFirstThing_policy

1 policy selected

Clicking on "Attach thing" will bring up the window illustrated below. Select your thing and click Attach.



Attach things to certificate(s)

Things will be attached to the following certificate(s):

e055c53465d3d5cdde69ba776ad585dfa81ff930a6a47bcd155b4d44d1a0ef7c

Choose one or more things

Search things

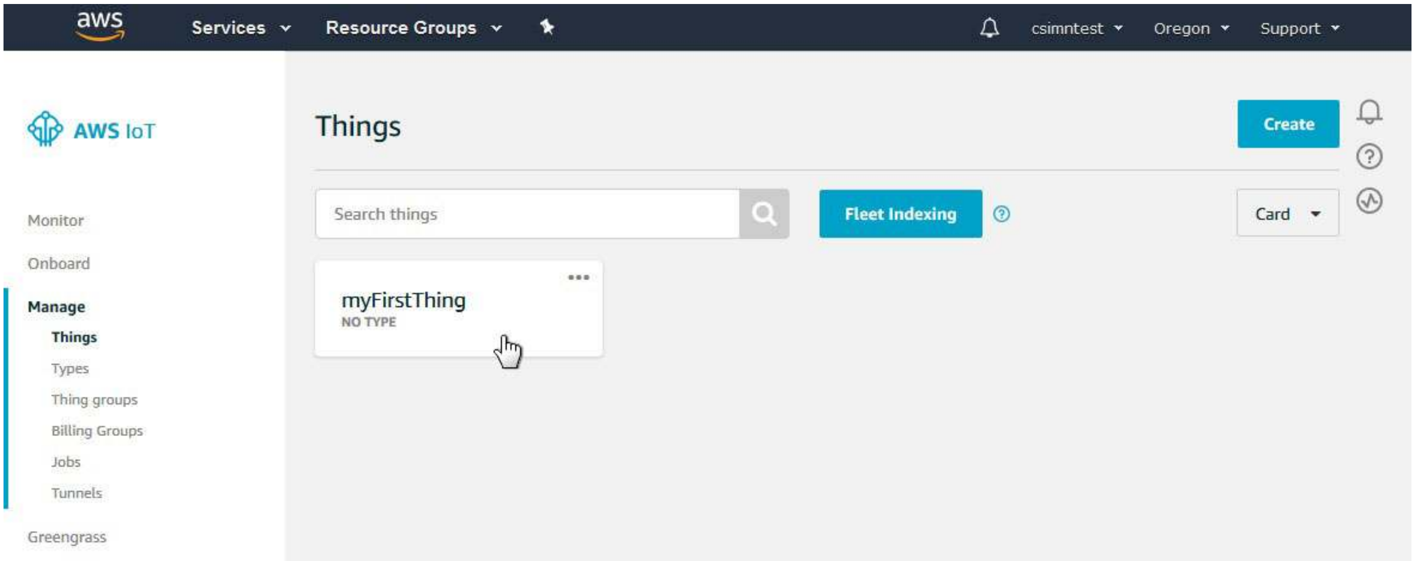
☑ myFirstThing

1 thing selected

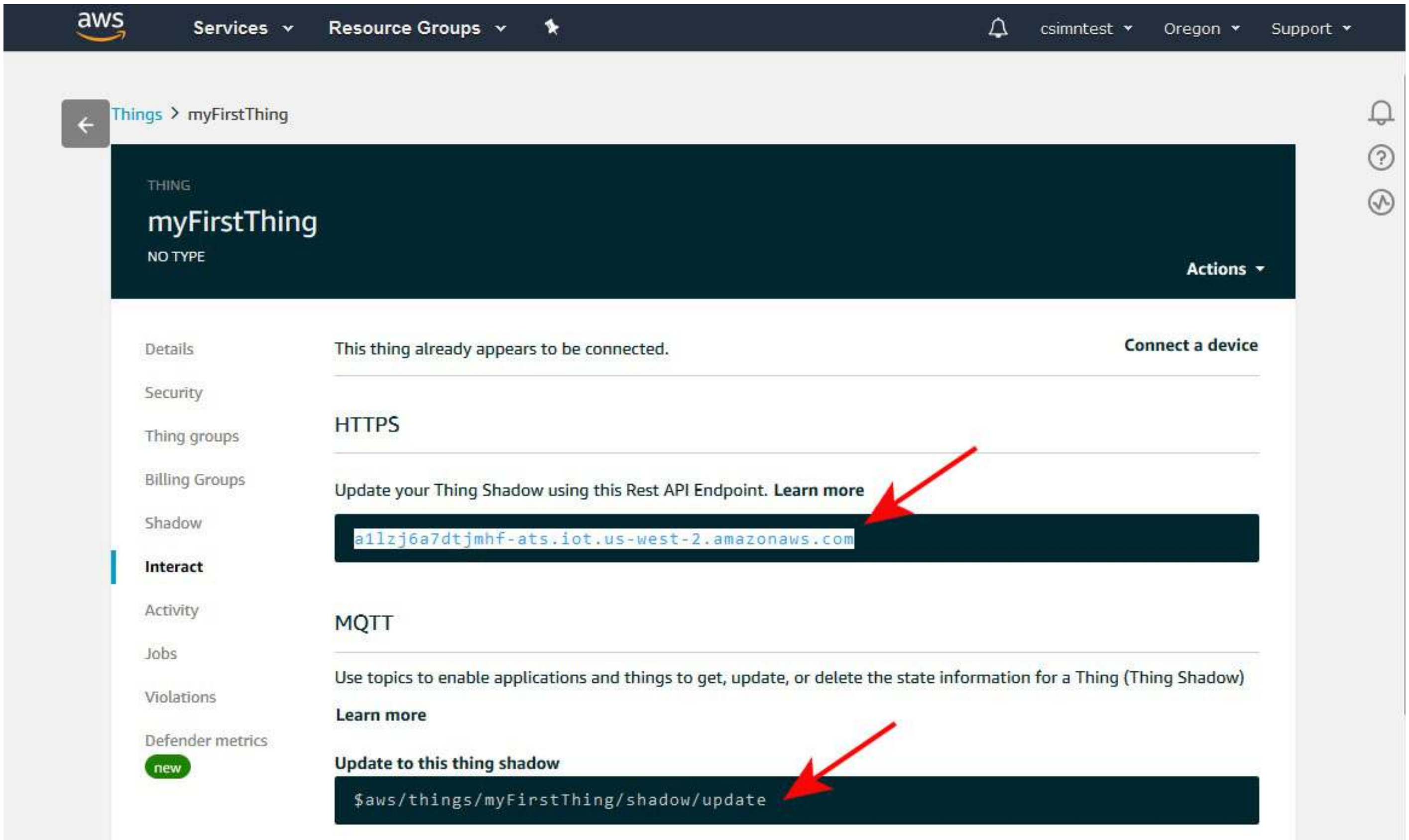
At this point, you are ready to move on to configuring the IoT Gateway.

14.4 Configure IoT Gateway

To begin the process of configuring the IoT Gateway, you need to look up some information about your thing on the AWS server. Go to the AWS IoT Manage Things page, and click on your Thing.



From the thing menu, select Interact. Copy the Rest API Endpoint - this will be entered as the host name in the IoT Gateway. Also make a note of the MQTT topic for updating this thing's shadow. This default topic should get created automatically for you by the IoT Gateway when you configure your device name.



This thing already appears to be connected.

HTTPS

Update your Thing Shadow using this Rest API Endpoint. **Learn more**

```
a1lzj6a7dtjmhf-ats.iot.us-west-2.amazonaws.com
```

MQTT

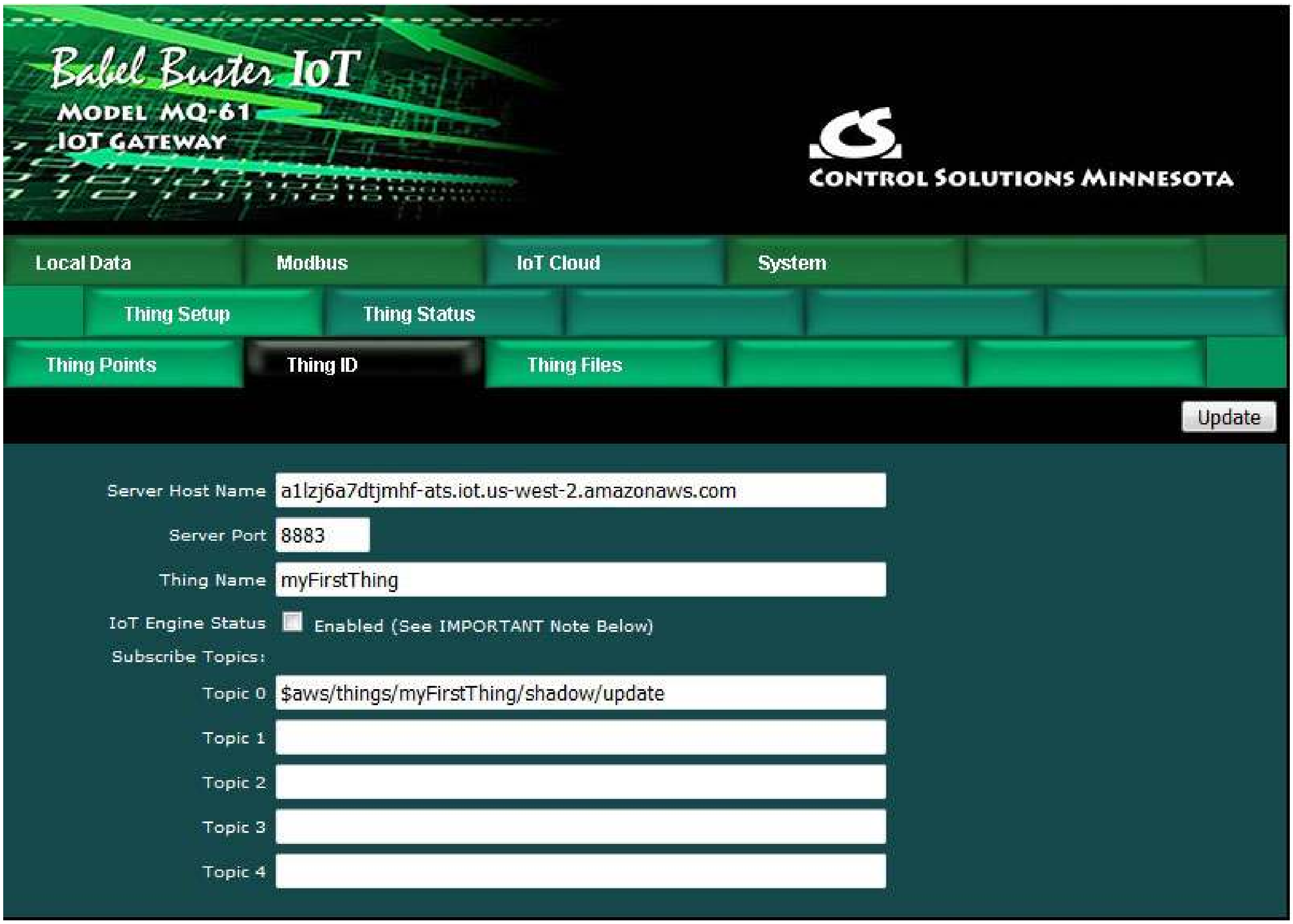
Use topics to enable applications and things to get, update, or delete the state information for a Thing (Thing Shadow)

Learn more

Update to this thing shadow

```
$aws/things/myFirstThing/shadow/update
```

Enter the Rest API Endpoint in the Server Host Name window. Set the server port to 8883. Enter the Thing Name exactly as it appeared on the AWS IoT Thing screen. Subscribe Topic 0 will update to show the default topic that includes your thing name once the name is entered.



Server Host Name a1lzj6a7dtjmhf-ats.iot.us-west-2.amazonaws.com

Server Port 8883

Thing Name myFirstThing

IoT Engine Status □ Enabled (See IMPORTANT Note Below)

Subscribe Topics:

Topic 0 $aws/things/myFirstThing/shadow/update

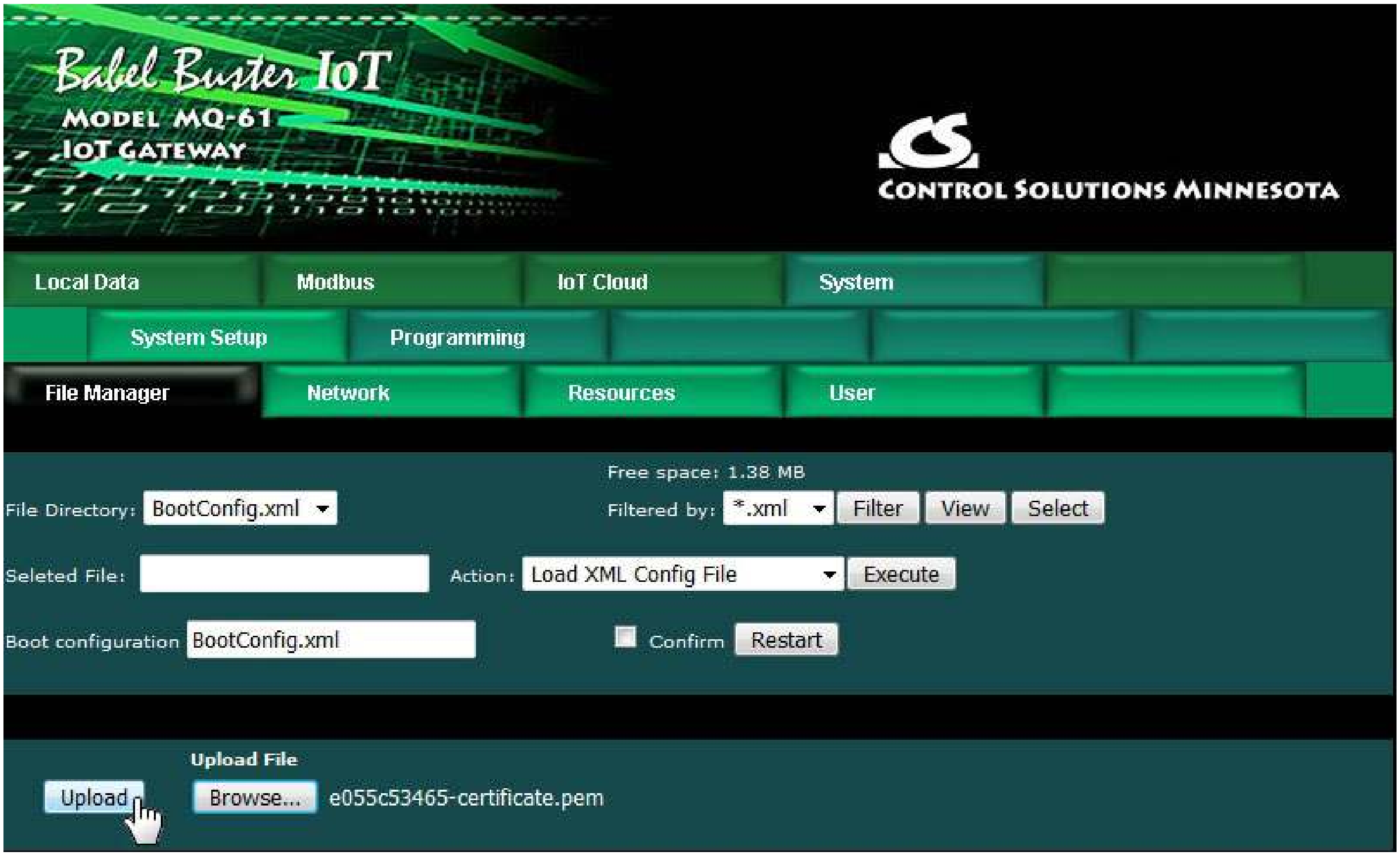
Topic 1

Topic 2

Topic 3

Topic 4

Go to the File Manager and upload your *.pem files - there should be three of them.



Free space: 1.38 MB

File Directory: BootConfig.xml

Filtered by: *.xml

Seleted File:

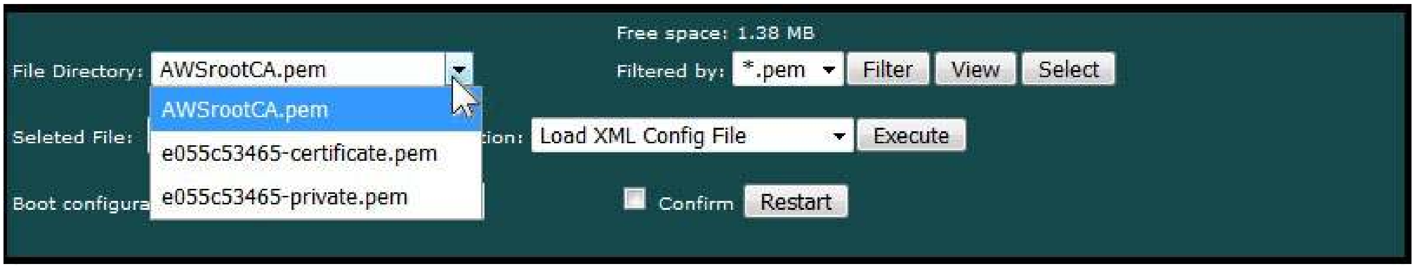
Action: Load XML Config File

Boot configuration BootConfig.xml

Upload File

e055c53465-certificate.pem

Confirm that you have uploaded the files by selecting *.pem from the filter list and then clicking Filter.



Free space: 1.38 MB

File Directory: AWSrootCA.pem

Filtered by: *.pem

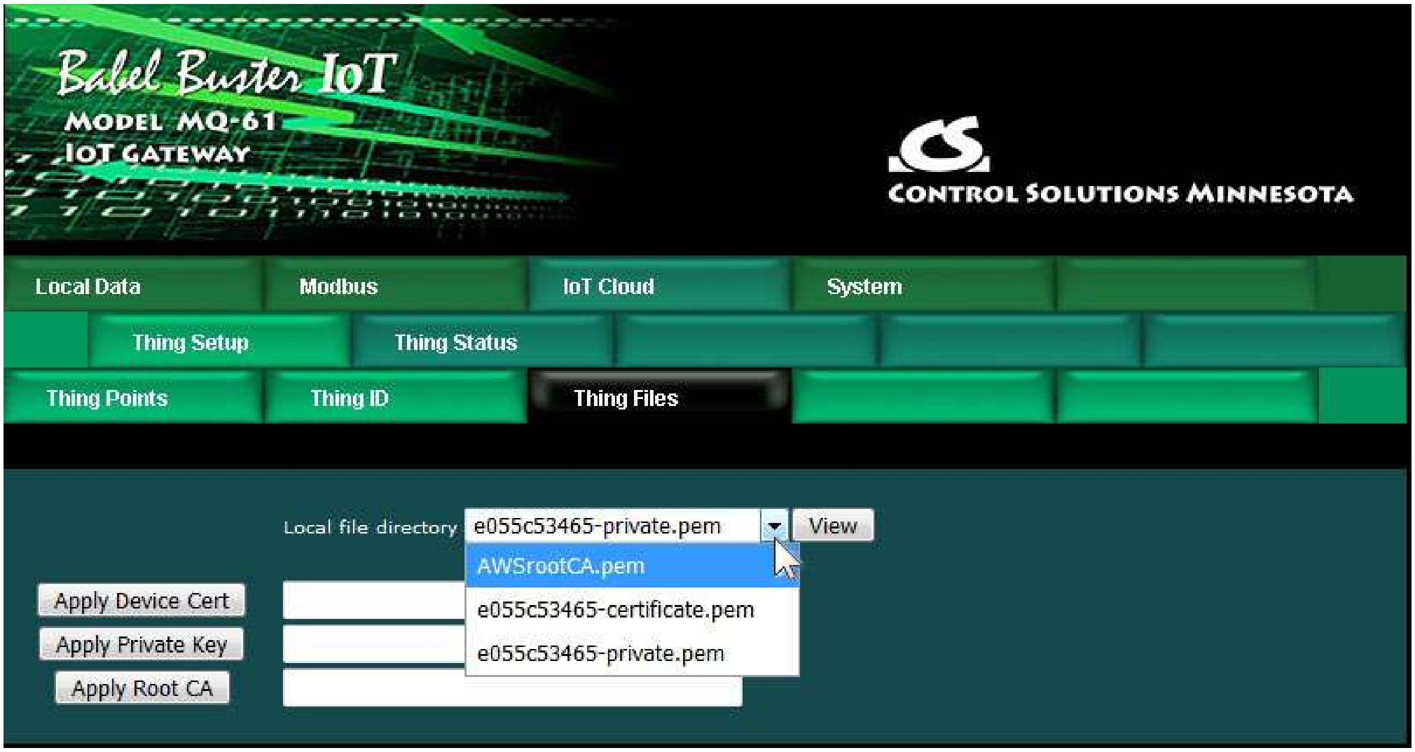
AWSrootCA.pem
e055c53465-certificate.pem
e055c53465-private.pem

Seleted File:

Action: Load XML Config File

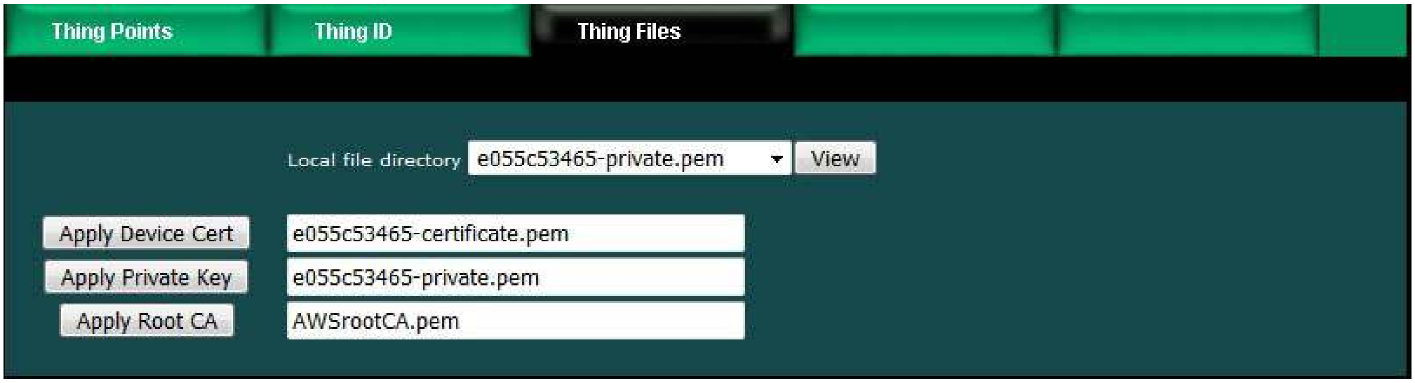
Boot configura

Go to the Thing Files page and select the *.pem files and apply them to the applicable position on the list.



Local file directory: e055c53465-private.pem

Your Thing Files page should end up looking like this.



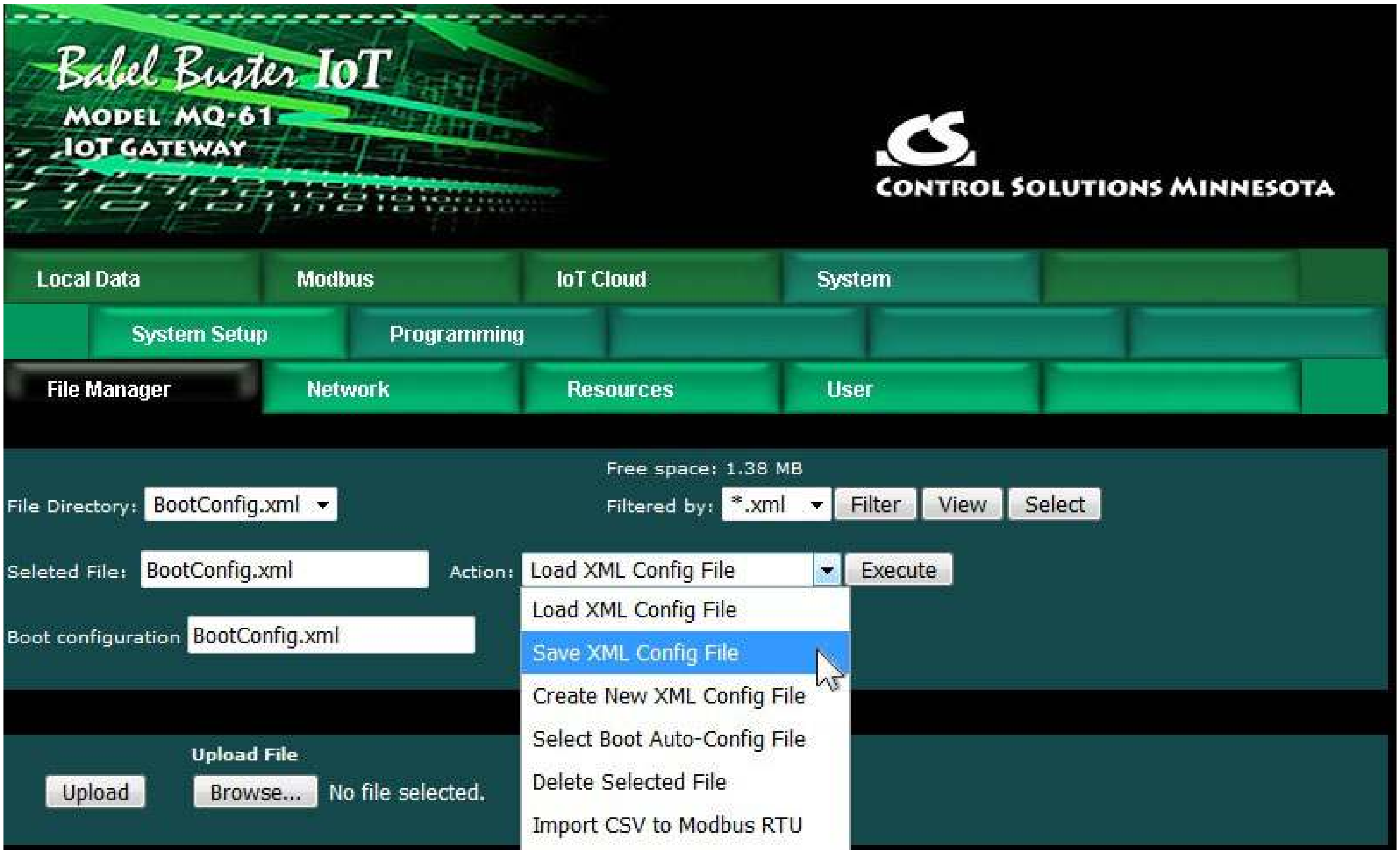
Local file directory: e055c53465-private.pem

Apply Device Cert: e055c53465-certificate.pem

Apply Private Key: e055c53465-private.pem

Apply Root CA: AWSrootCA.pem

IMPORTANT: Once you have entered the host name, thing name, and selected your SSL certificates, you should back to the File Manager and save your configuration to the BootConfig.xml file (or other XML file you create). If you skip this step, your changes will be lost the next time the IoT Gateway restarts or is power cycled.



Free space: 1.38 MB

File Directory: BootConfig.xml

Filtered by: *.xml

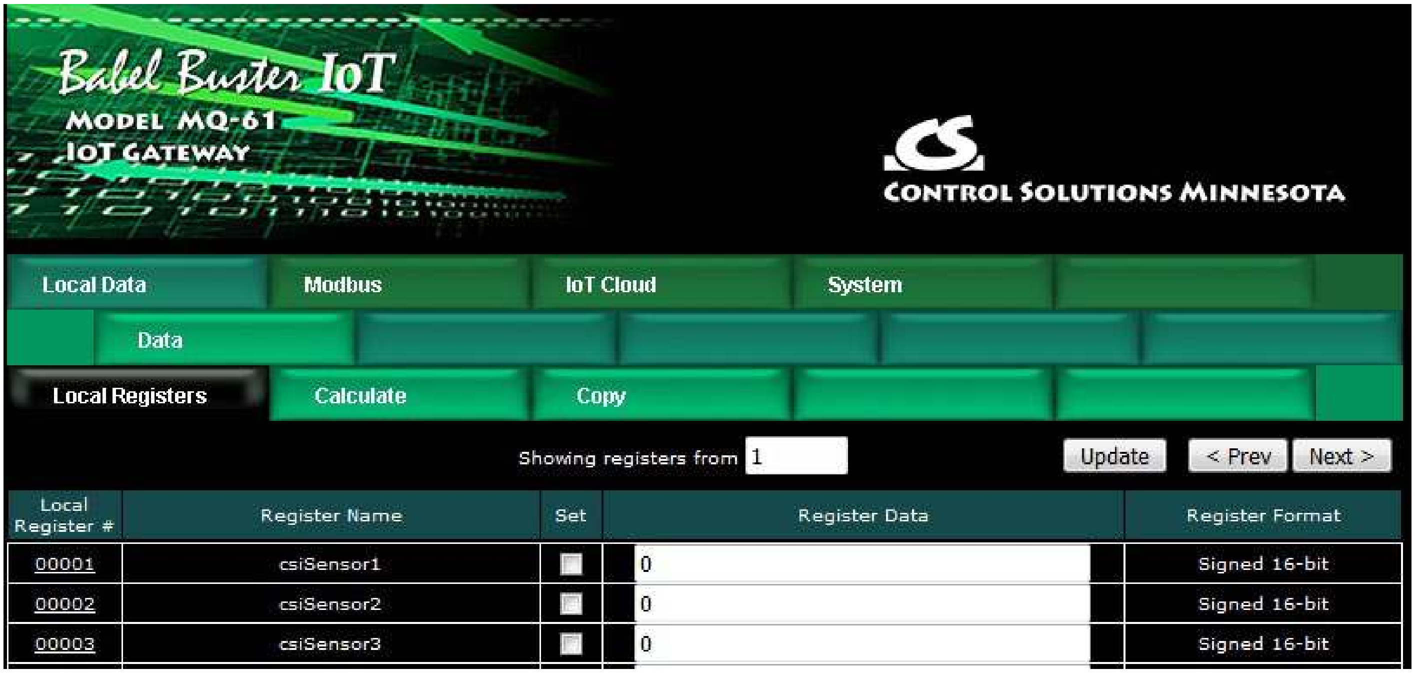
Seleted File: BootConfig.xml

Action: Load XML Config File

Boot configuration BootConfig.xml

No file selected.

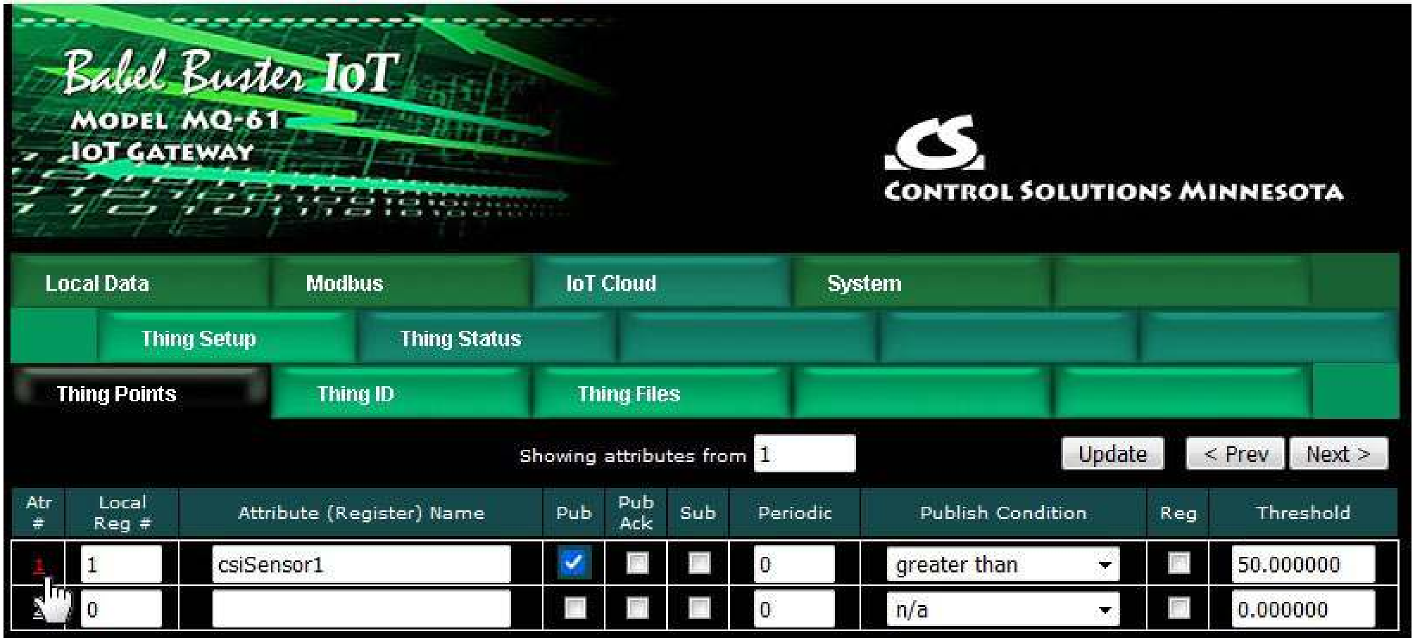
The overall Thing connection has now been set up. The only thing remaining is to tell the IoT Gateway which data points you want to publish to the AWS server. Create some local registers if you haven't already (see Section 4).



Showing registers from 1

Local Register #	Register Name	Set	Register Data	Register Format
00001	csiSensor1	□	0	Signed 16-bit
00002	csiSensor2	□	0	Signed 16-bit
00003	csiSensor3	□	0	Signed 16-bit

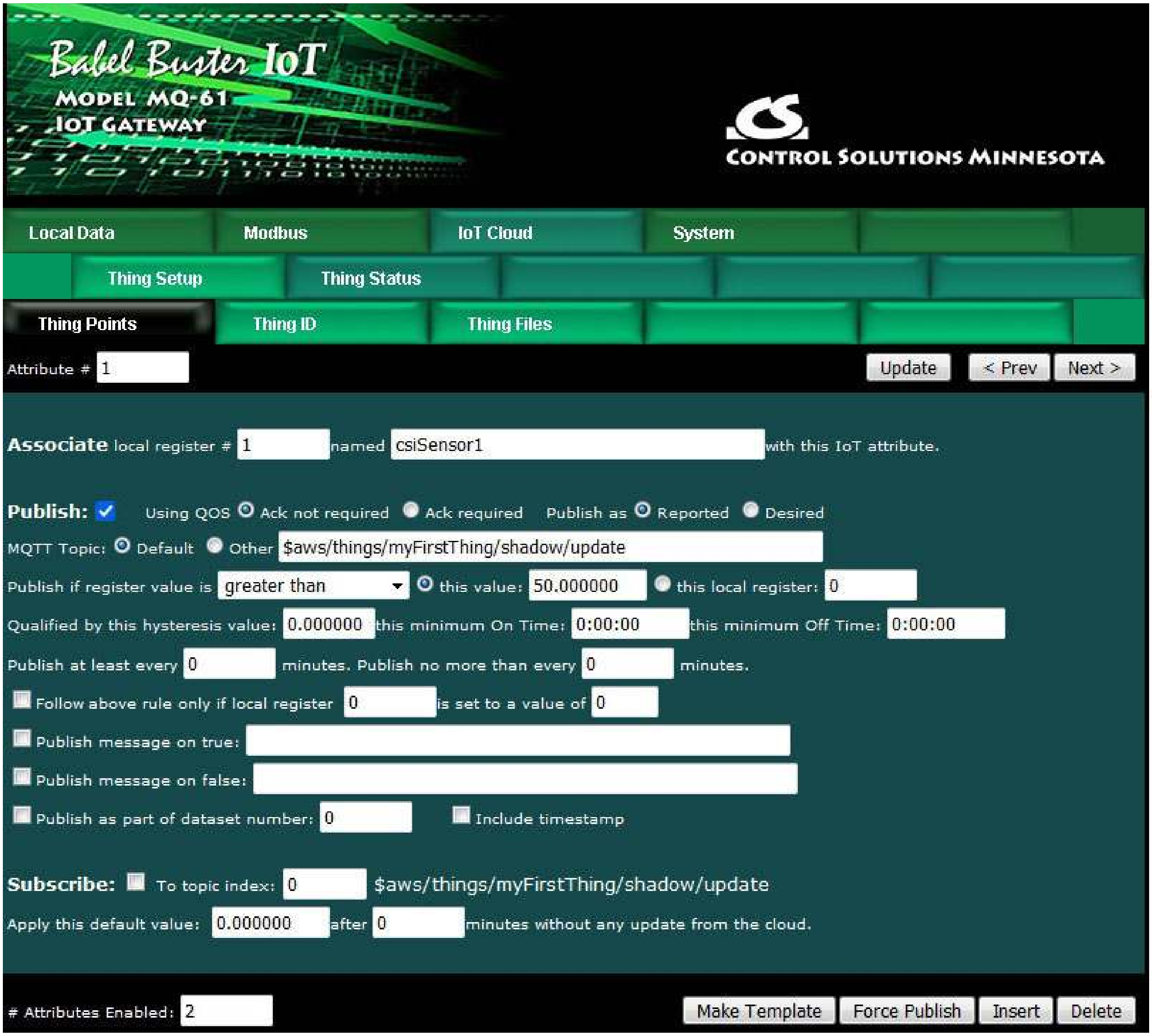
Now go to the Thing Points page. In the illustration below, we have already created a single point by making entries and selections on this Thing Points page as outlined at the very end of Section 13.1. To see the full publish/subscribe rule, click on the attribute number in the first column.



Showing attributes from 1

Atr #	Local Reg #	Attribute (Register) Name	Pub	Pub Ack	Sub	Periodic	Publish Condition	Reg	Threshold
1	1	csiSensor1	☑	□	□	0	greater than	□	50.000000
2	0		□	□	□	0	n/a	□	0.000000

The expanded view of the rule is illustrated below. This rule contains all of the minimum parameters to test publishing to the AWS server.



Attribute # 1

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 $aws/things/myFirstThing/shadow/update

Publish if register value is greater than ● this value: 50.000000 ○ 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

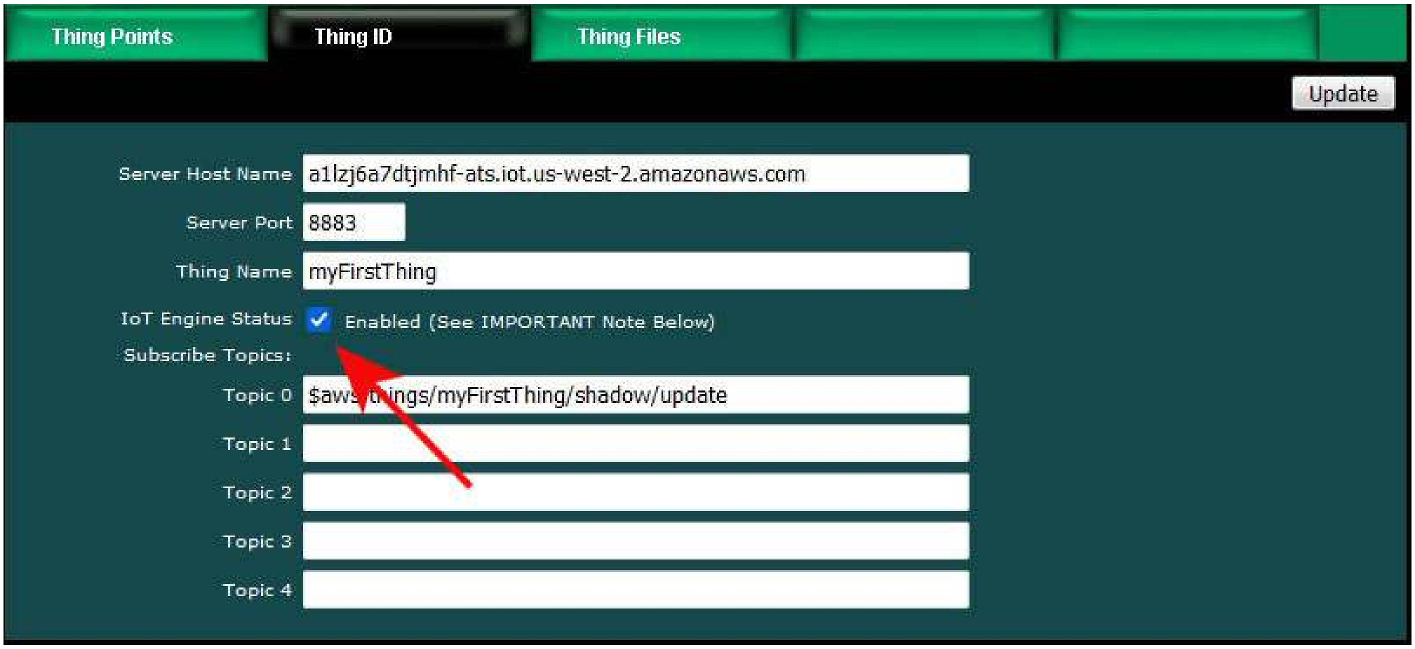
Subscribe: □ To topic index: 0 $aws/things/myFirstThing/shadow/update

Apply this default value: 0.000000 after 0 minutes without any update from the cloud.

Attributes Enabled: 2

14.5 Test Publish to Device Shadow

Now that everything else is all set up, go ahead and enable the IoT Engine.



Server Host Name a1lzj6a7dtjmhf-ats.iot.us-west-2.amazonaws.com

Server Port 8883

Thing Name myFirstThing

IoT Engine Status ☑ Enabled (See IMPORTANT Note Below)

Subscribe Topics:

Topic 0 $aws/things/myFirstThing/shadow/update

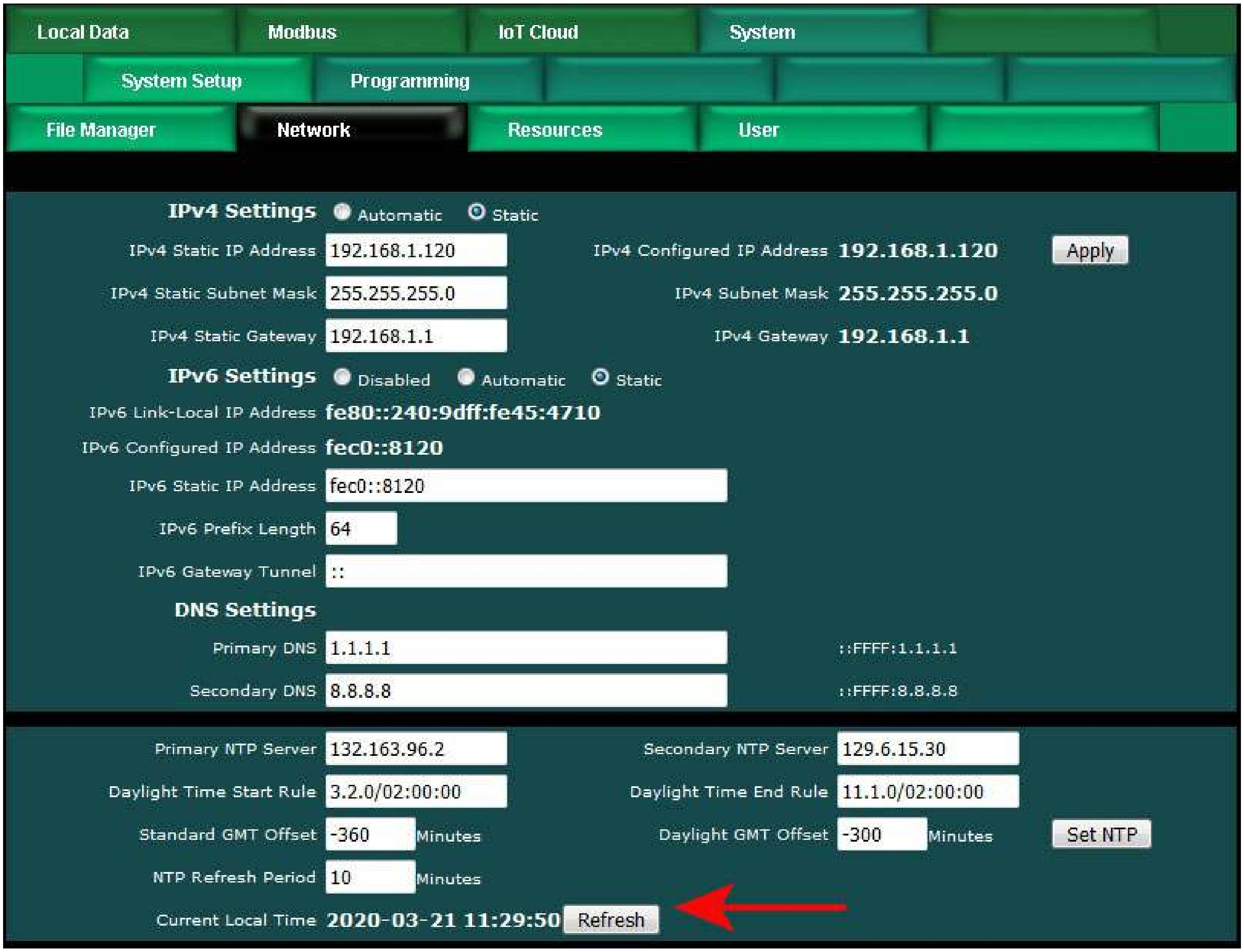
Topic 1

Topic 2

Topic 3

Topic 4

Make sure the NTP server has found the correct local time and date. The AWS server will not allow a secure connection if local time is not set correctly.



IPv4 Settings ○ Automatic ● Static

IPv4 Static IP Address 192.168.1.120

IPv4 Configured IP Address **192.168.1.120**

IPv4 Static Subnet Mask 255.255.255.0

IPv4 Subnet Mask **255.255.255.0**

IPv4 Static Gateway 192.168.1.1

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::8120**

IPv6 Static IP Address fec0::8120

IPv6 Prefix Length 64

IPv6 Gateway Tunnel ::

DNS Settings

Primary DNS 1.1.1.1 ::FFFF:1.1.1.1

Secondary DNS 8.8.8.8 ::FFFF:8.8.8.8

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

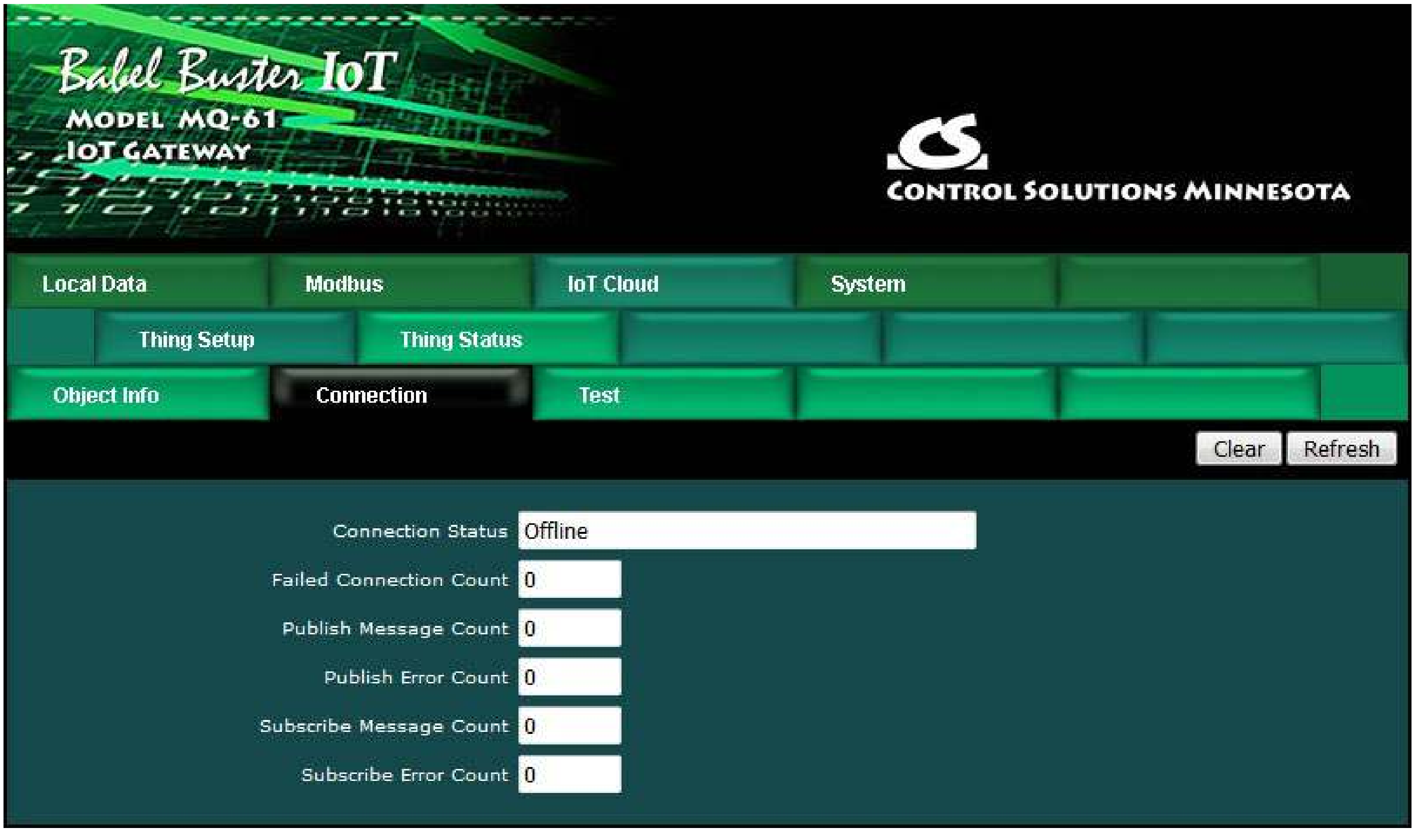
Standard GMT Offset -360 Minutes

Daylight GMT Offset -300 Minutes

NTP Refresh Period 10 Minutes

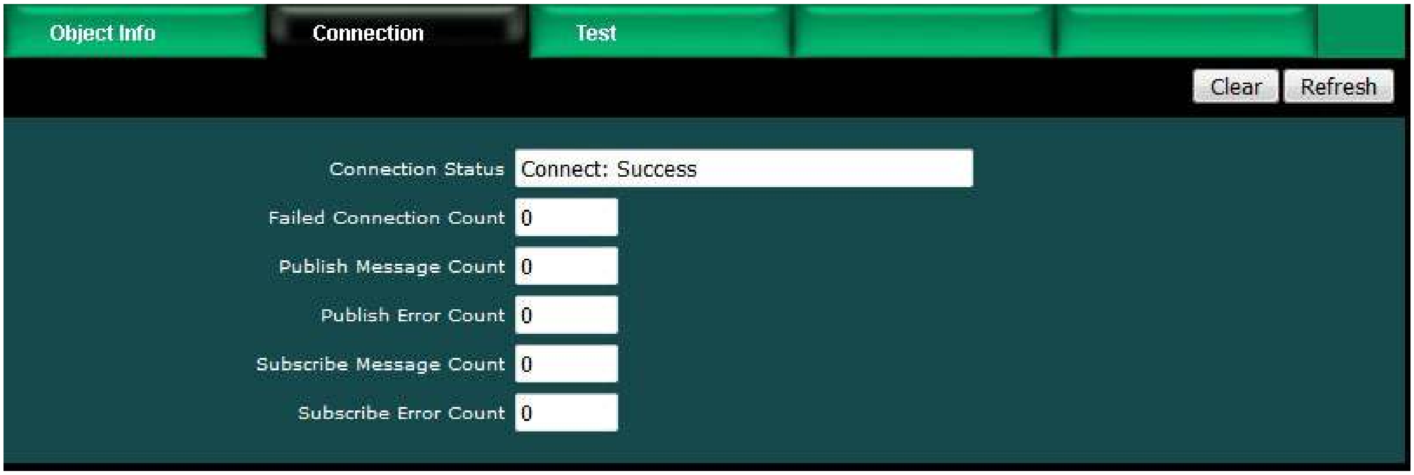
Current Local Time **2020-03-21 11:29:50**

Initially, the status will simply be "Offline".



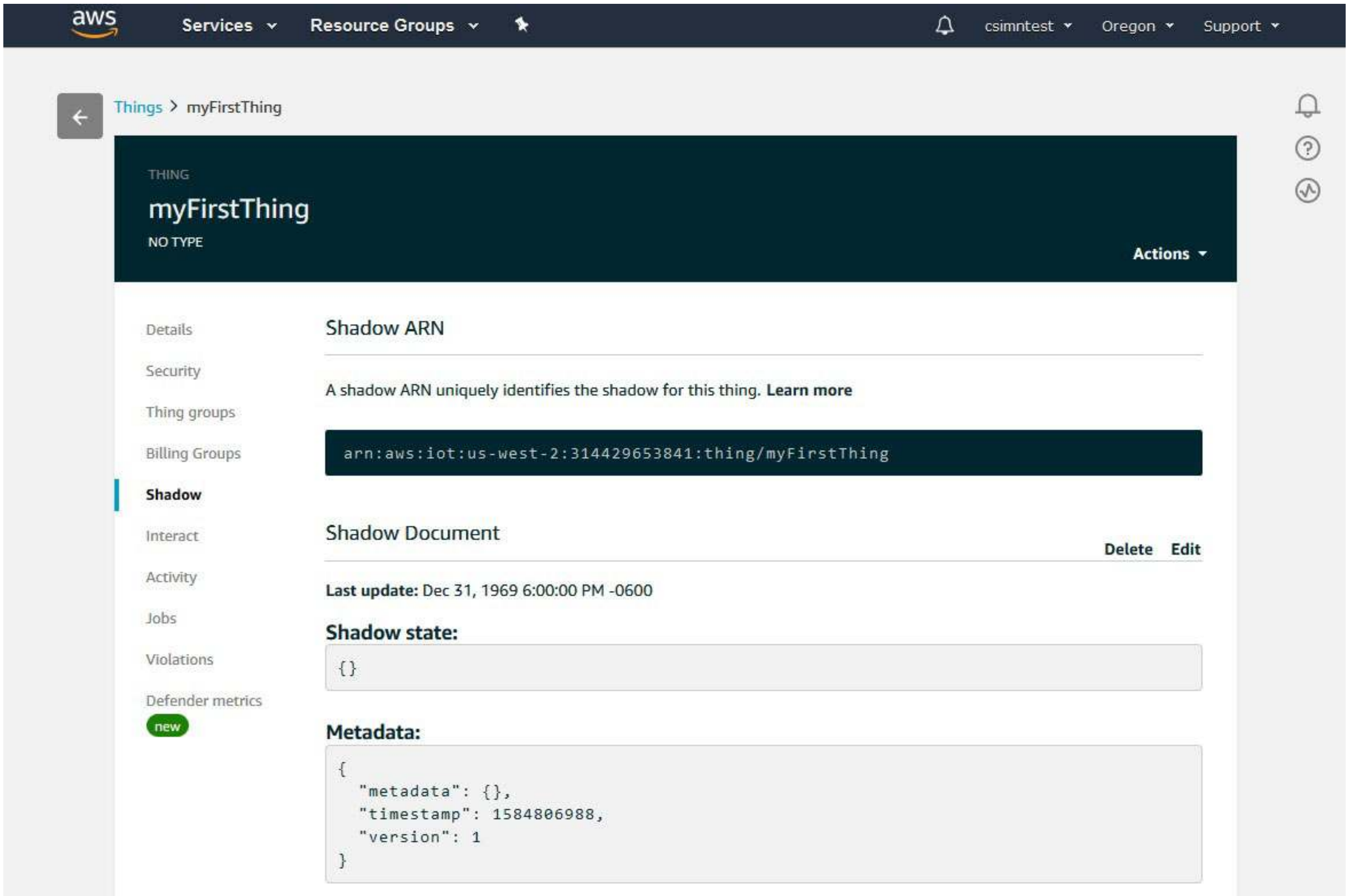
Connection Status Offline
Failed Connection Count 0
Publish Message Count 0
Publish Error Count 0
Subscribe Message Count 0
Subscribe Error Count 0

After a delay of several seconds, you will hopefully see "Connect: Success". If the status says something about unknown host or anything to the effect of being unable to connect, be sure you have a DNS server accessible that can look up Amazon's IP address. If it says anything about SSL certificate error, recheck your SSL certificates. Make sure you are using the correct certificates, assigned to the correct function on the Thing Files page. Also do note that any time you create a new Thing, you will be generating a new set of certificates. If you recreated your Thing, you need to download the new certificates for the new Thing.



Connection Status Connect: Success
Failed Connection Count 0
Publish Message Count 0
Publish Error Count 0
Subscribe Message Count 0
Subscribe Error Count 0

The default publish topic is to update the device shadow on the AWS server. Go to your Thing menu, and display its Shadow. You will initially see that nothing is contained within the Shadow state.



Shadow ARN

A shadow ARN uniquely identifies the shadow for this thing. **Learn more**

```
arn:aws:iot:us-west-2:314429653841:thing/myFirstThing
```

Shadow Document

Last update: Dec 31, 1969 6:00:00 PM -0600

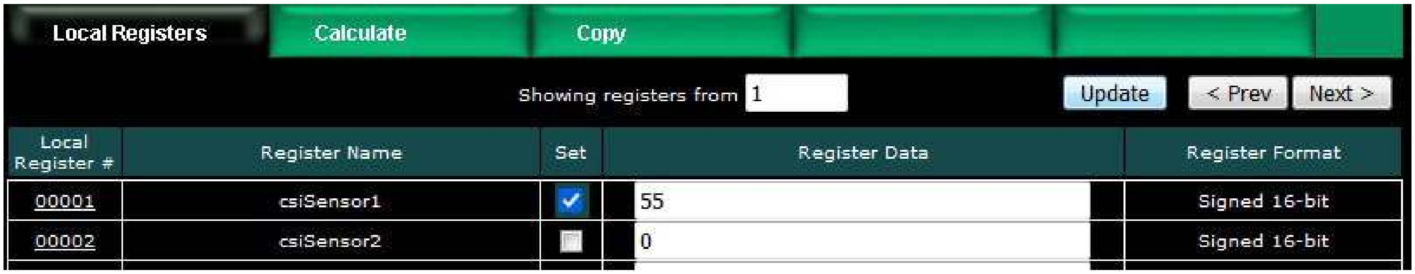
Shadow state:

```
{}
```

Metadata:

```
{
  "metadata": {},
  "timestamp": 1584806988,
  "version": 1
}
```

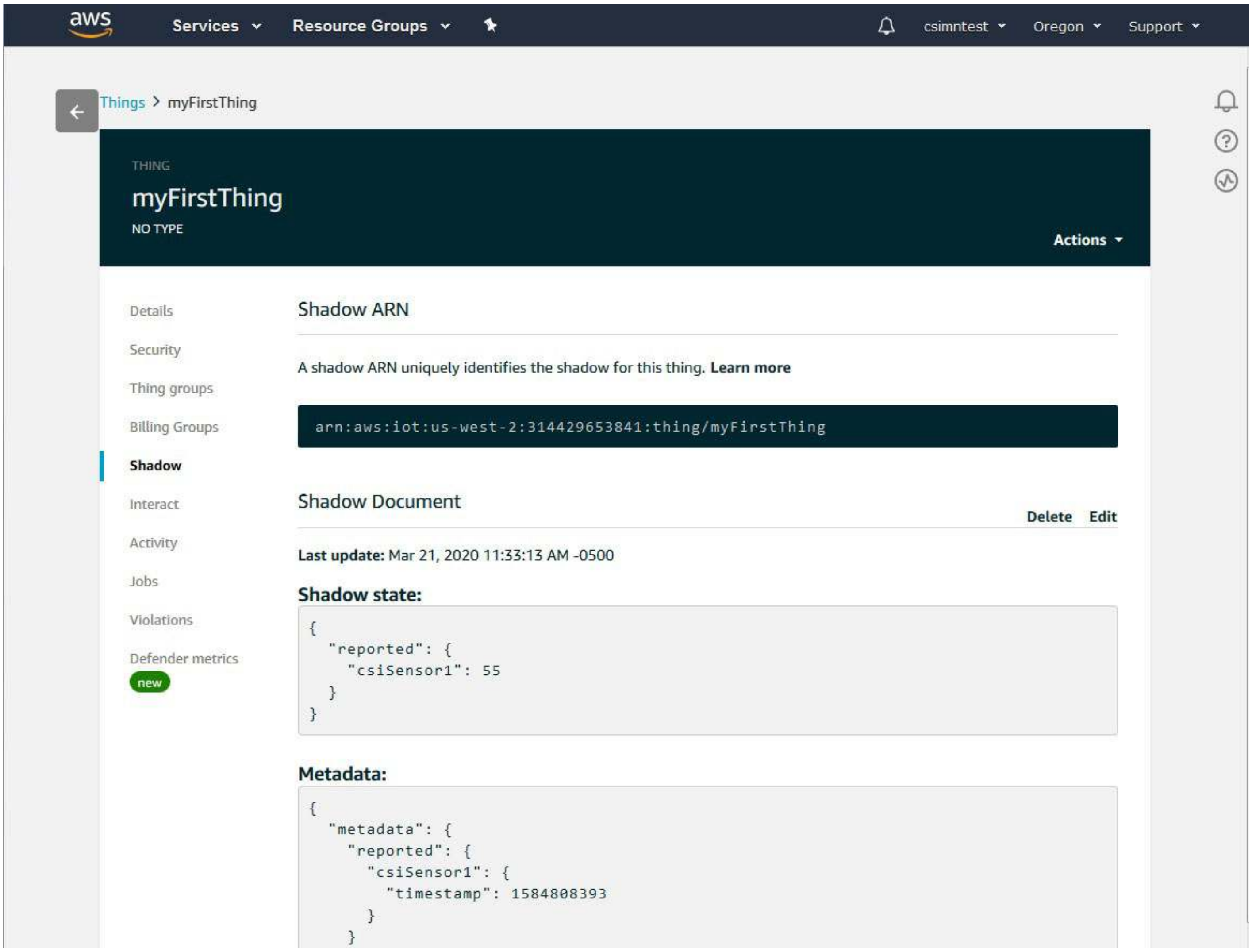
The rule we set up above says to publish this data point any time its value exceeds 50. So go ahead and force the value to something greater than 50 (this assumes you do not have any Modbus maps configured yet that will overwrite this register while attempting this test).



Showing registers from 1

Local Register #	Register Name	Set	Register Data	Register Format
00001	csiSensor1	☑	55	Signed 16-bit
00002	csiSensor2	□	0	Signed 16-bit

If successful, the result will be posted in the Shadow almost immediately. If you see this, congratulations, you have now published your first piece of data to the AWS server using your Thing on the Internet of Things.



Shadow ARN

A shadow ARN uniquely identifies the shadow for this thing. **Learn more**

```
arn:aws:iot:us-west-2:314429653841:thing/myFirstThing
```

Shadow Document

Last update: Mar 21, 2020 11:33:13 AM -0500

Shadow state:

```
{
  "reported": {
    "csiSensor1": 55
  }
}
```

Metadata:

```
{
  "metadata": {
    "reported": {
      "csiSensor1": {
        "timestamp": 1584808393
      }
    }
```

Babel Buster IoT
MODEL MQ-61
MODEL BB3-6101-MQ
IOT GATEWAY

CONTROL SOLUTIONS MINNESOTA

15. Configuring IoT Gateway to Subscribe to AWS

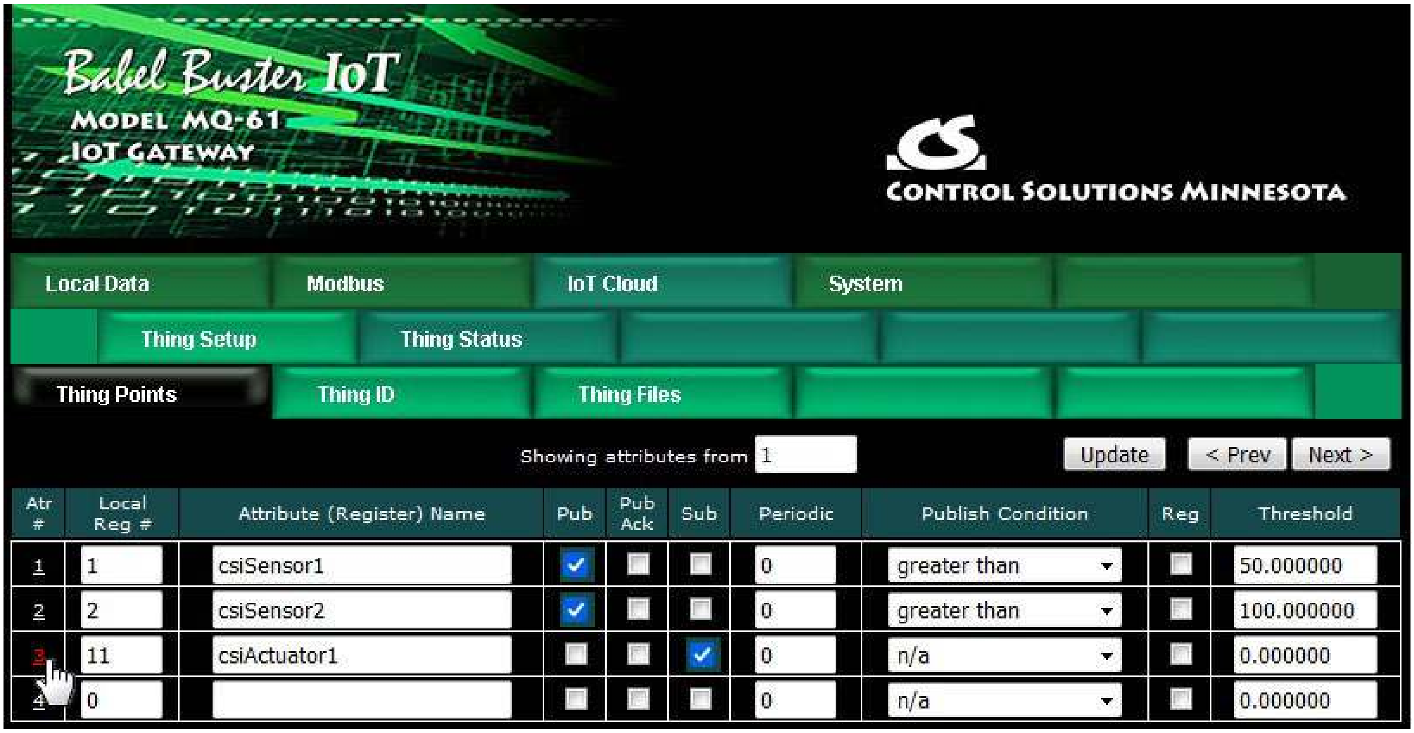
The MQTT term "Subscribe", from a controls perspective, would be most closely associated with the action of an actuator. You want to control an output based on setpoint data received from some external source. In the instance we are working with here, we are Subscribing to data from the AWS server.

The IoT Gateway 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 IoT Gateway, 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.

While using the Shadow object is not required, the benefit of doing so is that the desired value will still be available should the IoT Gateway be temporarily offline or disconnected. When the IoT Gateway reconnects, it will retrieve the desired state information from the Shadow object and set its local registers accordingly.

15.1 Configure IoT Gateway

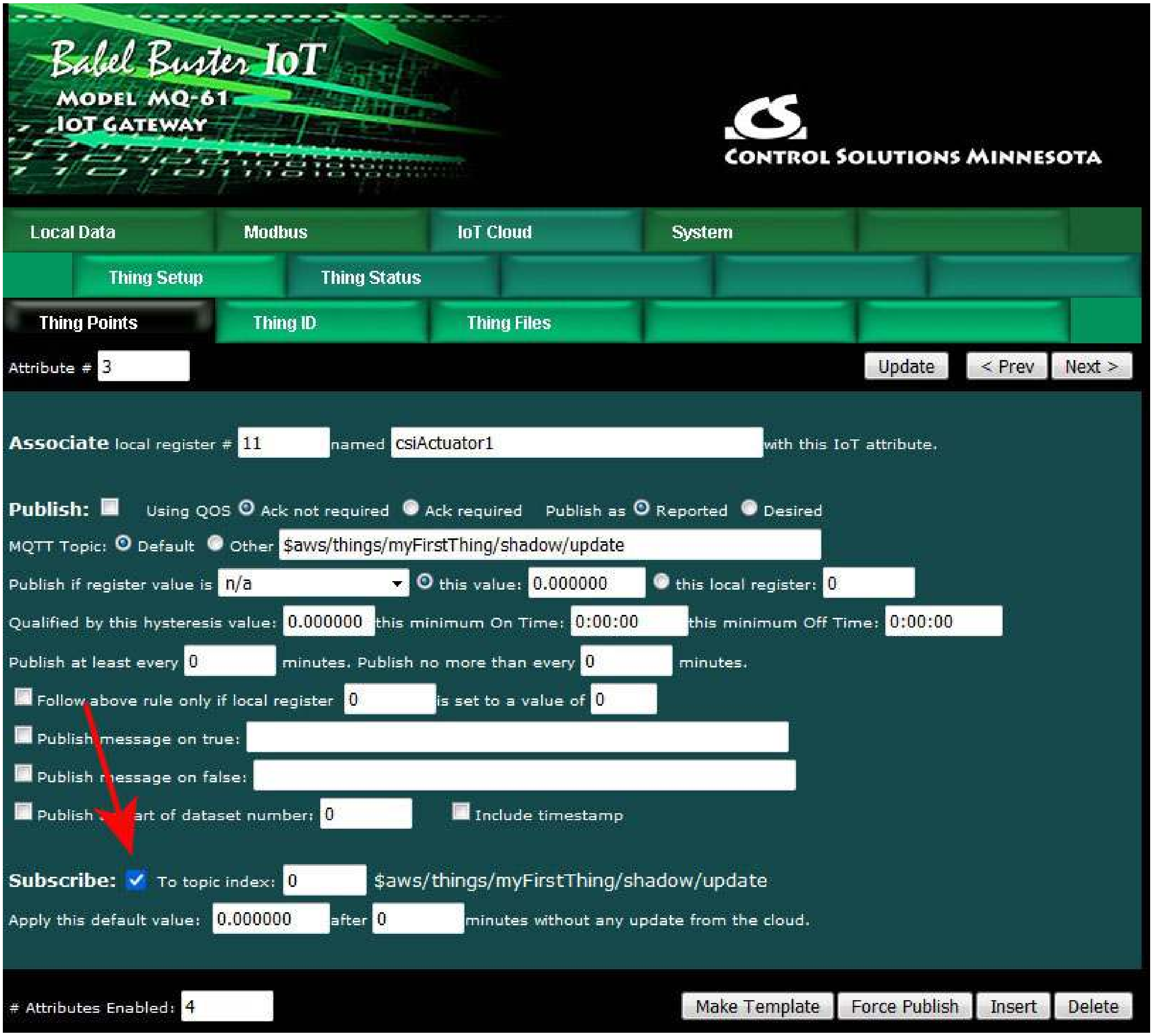
The simples form of a Subscribe point is illustrated below. To see the expanded view, click on the attribute number in the first column.



Showing attributes from 1

Atr #	Local Reg #	Attribute (Register) Name	Pub	Pub Ack	Sub	Periodic	Publish Condition	Reg	Threshold
1	1	csiSensor1	☑	□	□	0	greater than	□	50.000000
2	2	csiSensor2	☑	□	□	0	greater than	□	100.000000
3	11	csiActuator1	□	□	☑	0	n/a	□	0.000000
4	0		□	□	□	0	n/a	□	0.000000

You will most often just use the default subscribe topic (illustrated below for "myFirstThing"), so all you really need to do is select Subscribe using the checkbox. Then click Update.



Attribute # 3

Associate local register # 11 named csiActuator1 with this IoT attribute.

Publish: □ Using QOS ● Ack not required ○ Ack required Publish as ● Reported ○ Desired

MQTT Topic: ● Default ○ Other $aws/things/myFirstThing/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:

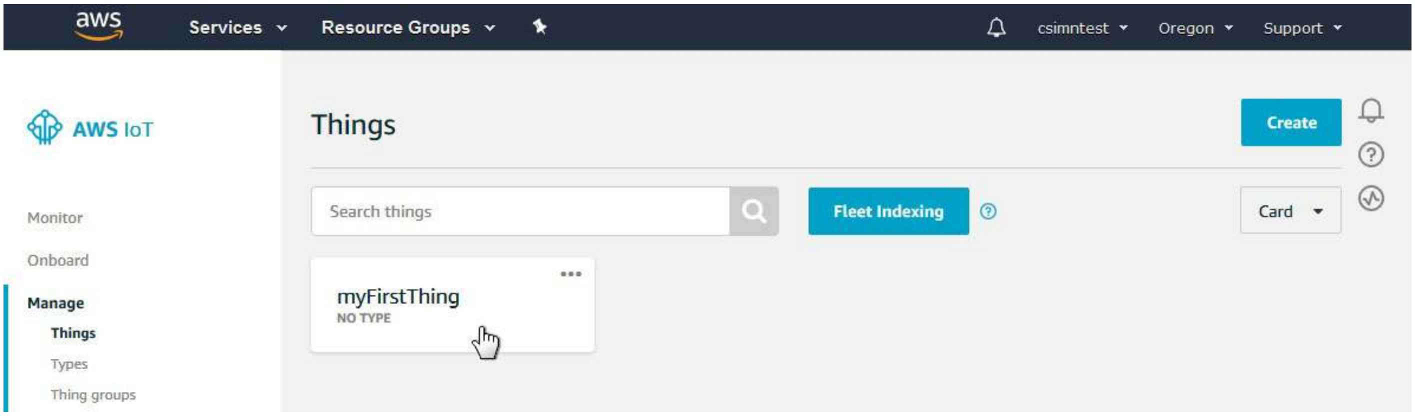
□ Publish as part of dataset number: 0 □ Include timestamp

Subscribe: ☑ To topic index: 0 $aws/things/myFirstThing/shadow/update

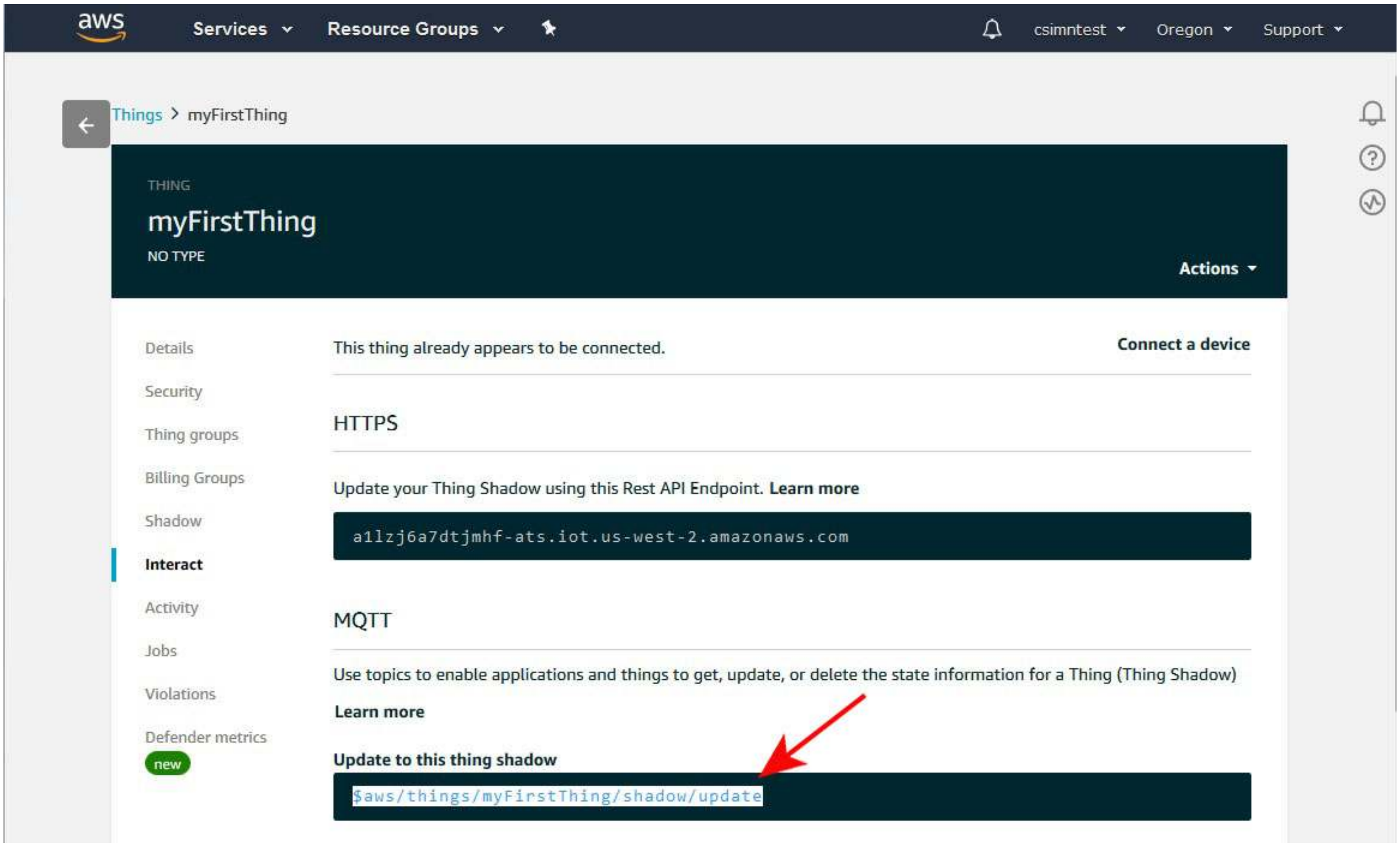
Apply this default value: 0.000000 after 0 minutes without any update from the cloud.

Attributes Enabled: 4

To test the subscription, go to your AWS IoT management console and select your Thing. If you have not already created a Thing as outlined in Section 14, go back to that section and create a Thing as outlined. Doing a test publish is recommended as this will confirm that the connection was successful.



You can verify the topic that you should subscribe to by clicking the Interact link on the menu.



This thing already appears to be connected.

HTTPS

Update your Thing Shadow using this Rest API Endpoint. Learn more

```
a1lzj6a7dtjmhf-ats.iot.us-west-2.amazonaws.com
```

MQTT

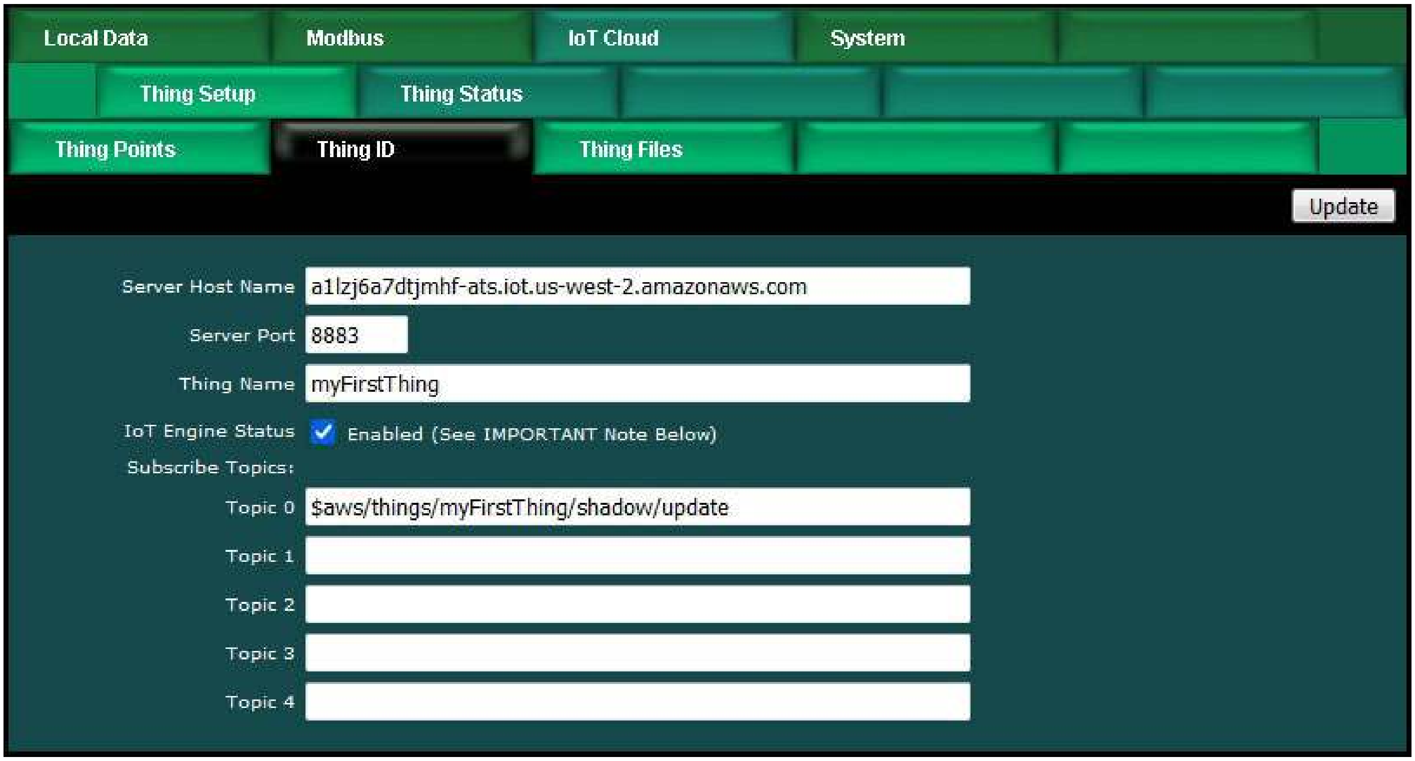
Use topics to enable applications and things to get, update, or delete the state information for a Thing (Thing Shadow)

Learn more

Update to this thing shadow

```
$aws/things/myFirstThing/shadow/update
```

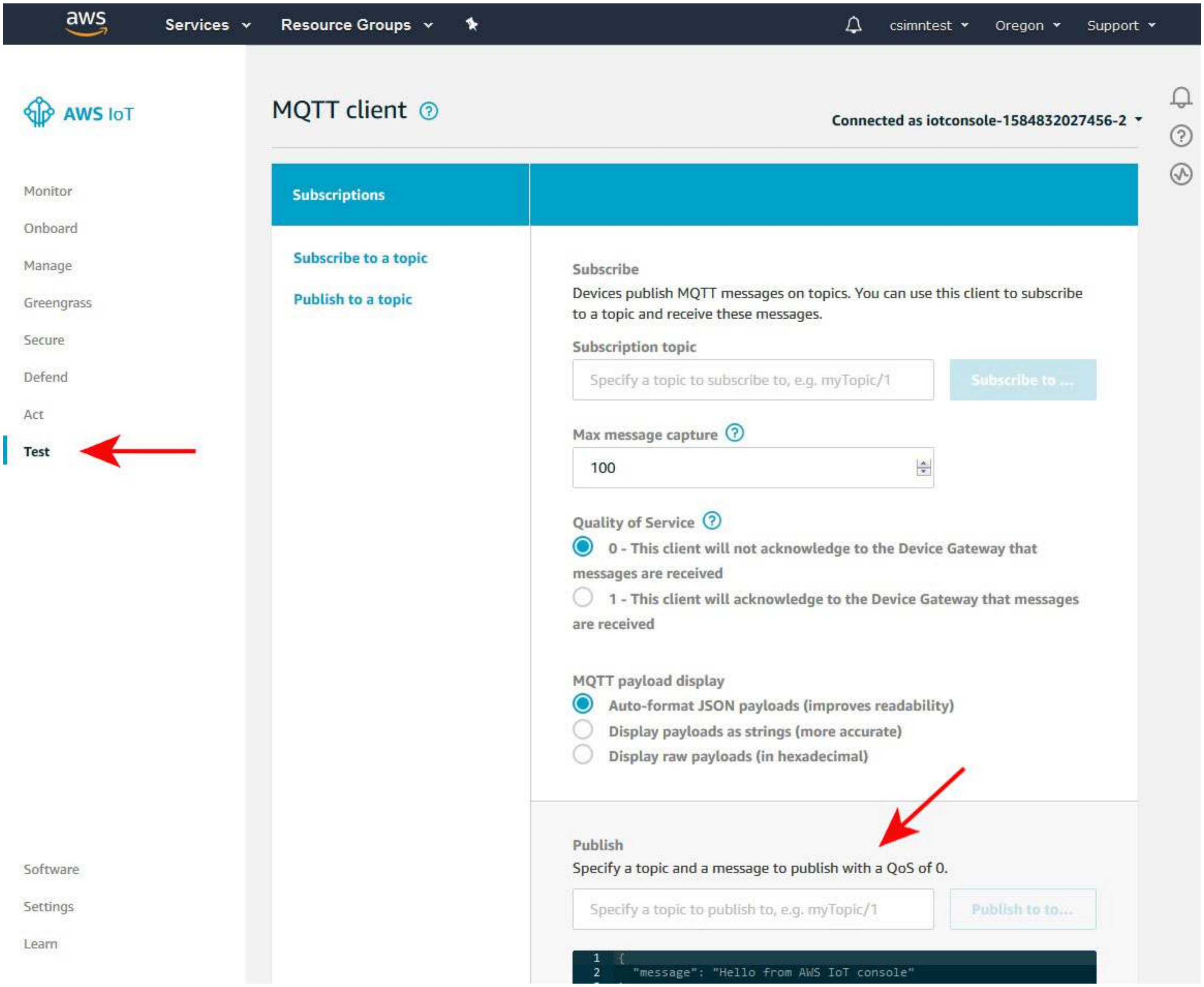
The default topic will normally be created for you when you enter the thing name.



Server Host Name	a1lzj6a7dtjmhf-ats.iot.us-west-2.amazonaws.com
Server Port	8883
Thing Name	myFirstThing
IoT Engine Status	☑ Enabled (See IMPORTANT Note Below)
Subscribe Topics:	
Topic 0	$aws/things/myFirstThing/shadow/update
Topic 1	
Topic 2	
Topic 3	
Topic 4	

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 click on the Test link. Using the MQTT client found here, you will manually publish to our default topic.



Subscribe

Devices publish MQTT messages on topics. You can use this client to subscribe to a topic and receive these messages.

Subscription topic: Specify a topic to subscribe to, e.g. myTopic/1

Max message capture: 100

Quality of Service

● 0 - This client will not acknowledge to the Device Gateway that messages are received

○ 1 - This client will acknowledge to the Device Gateway that messages are received

MQTT payload display

● Auto-format JSON payloads (improves readability)

○ Display payloads as strings (more accurate)

○ Display raw payloads (in hexadecimal)

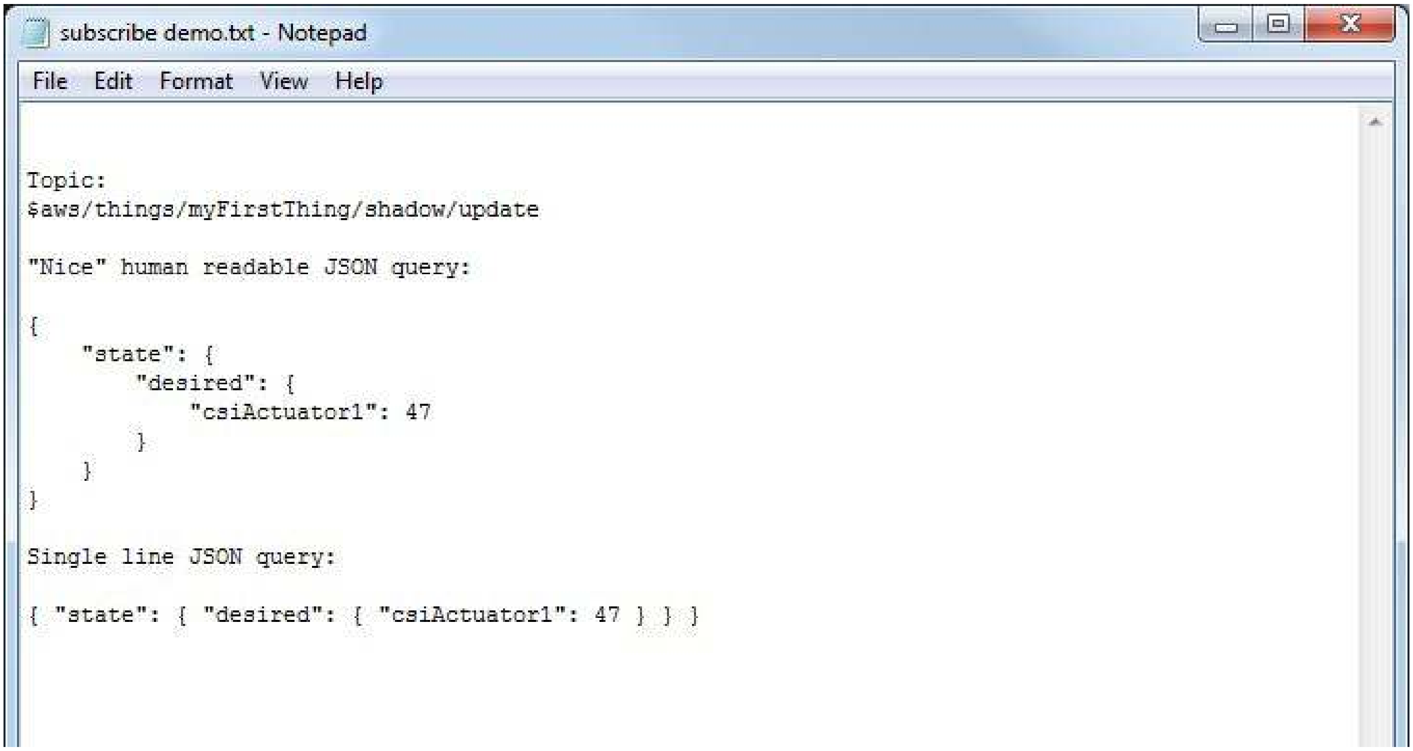
Publish

Specify a topic and a message to publish with a QoS of 0.

Specify a topic to publish to, e.g. myTopic/1

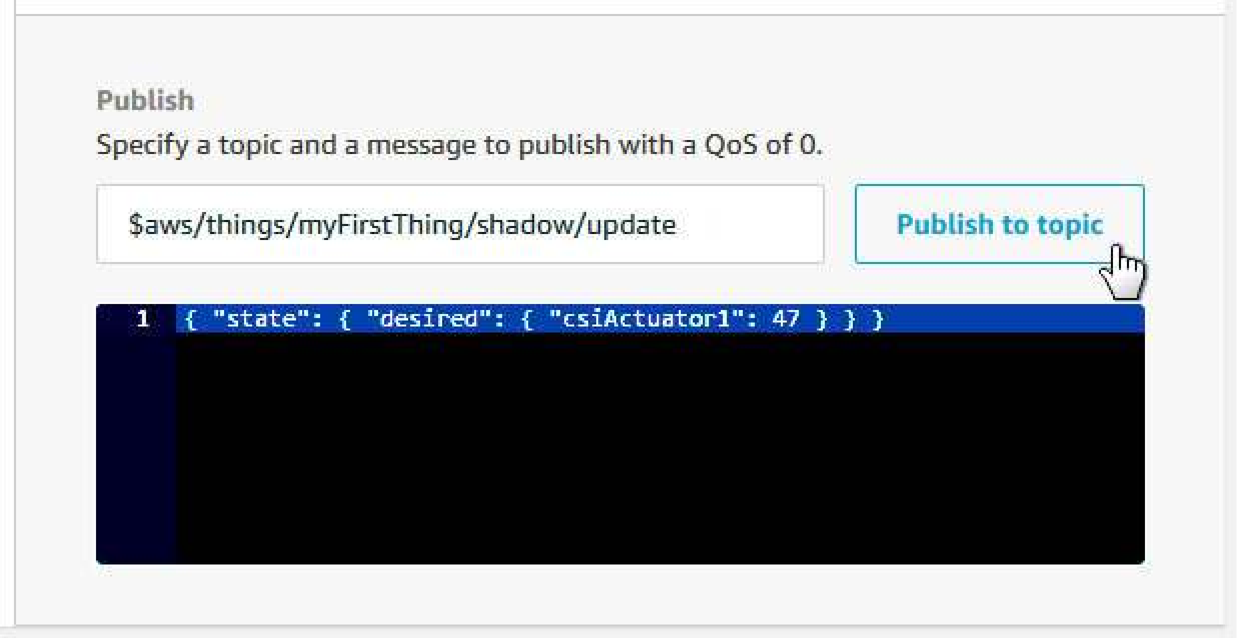
```
{
  "message": "Hello from AWS IoT console"
```

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/myFirstThing/shadow/update

"Nice" human readable JSON query:

{
    "state": {
        "desired": {
            "csiActuator1": 47
        }
    }
}

Single line JSON query:

{ "state": { "desired": { "csiActuator1": 47 } } }
```

In the Publish section of the MQTT Client page, enter the topic illustrated above, and the payload. The payload section does not actually appear as bright as illustrated here - this graphic was enhanced for readability as a document.



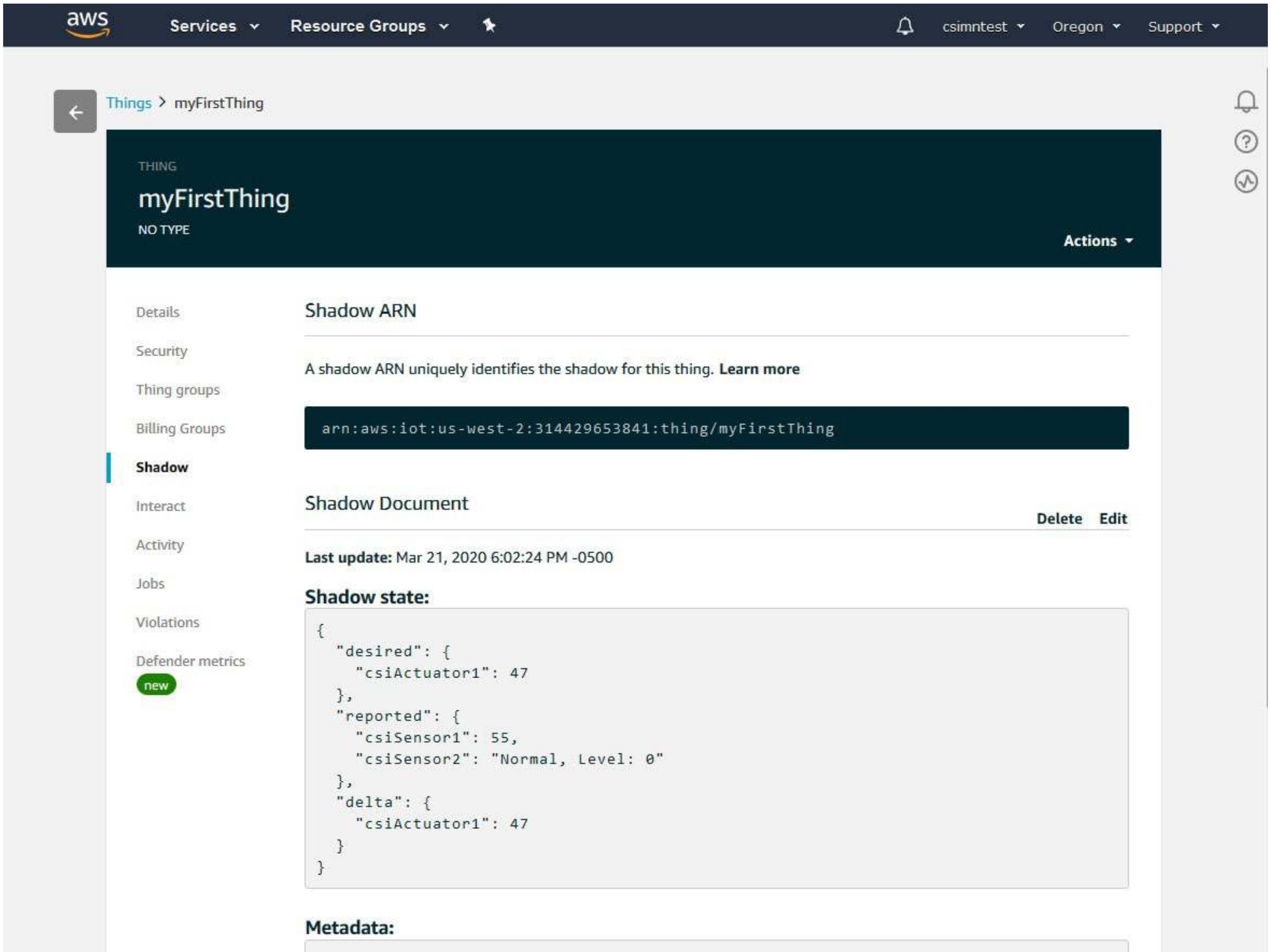
Publish

Specify a topic and a message to publish with a QoS of 0.

$aws/things/myFirstThing/shadow/update

```
{ "state": { "desired": { "csiActuator1": 47 } } }
```

If you go back and look at your thing's Shadow at this point you will see the "desired" state that was published.



Shadow ARN

A shadow ARN uniquely identifies the shadow for this thing. **Learn more**

```
arn:aws:iot:us-west-2:314429653841:thing/myFirstThing
```

Shadow Document

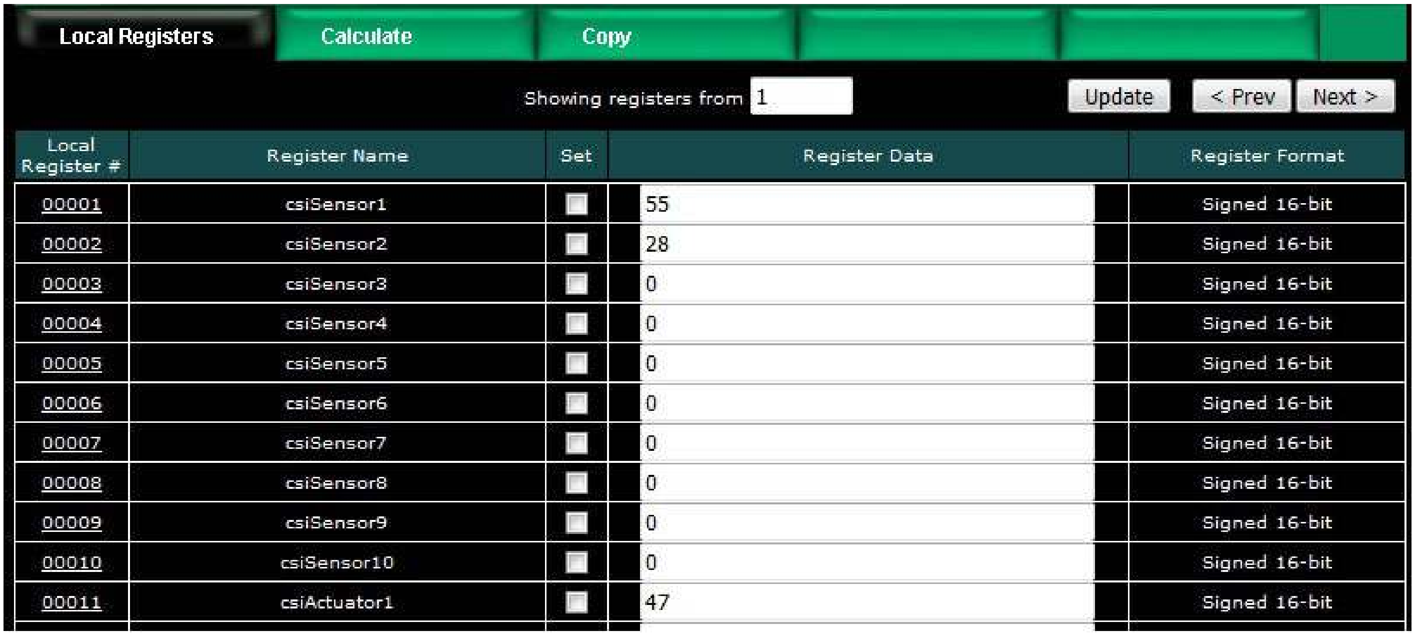
Last update: Mar 21, 2020 6:02:24 PM -0500

Shadow state:

```
{
  "desired": {
    "csiActuator1": 47
  },
  "reported": {
    "csiSensor1": 55,
    "csiSensor2": "Normal, Level: 0"
  },
  "delta": {
    "csiActuator1": 47
  }
}
```

Metadata:

You should now be able to go to the IoT Gateway's Local Registers page and see the value that you published from the MQTT Client.



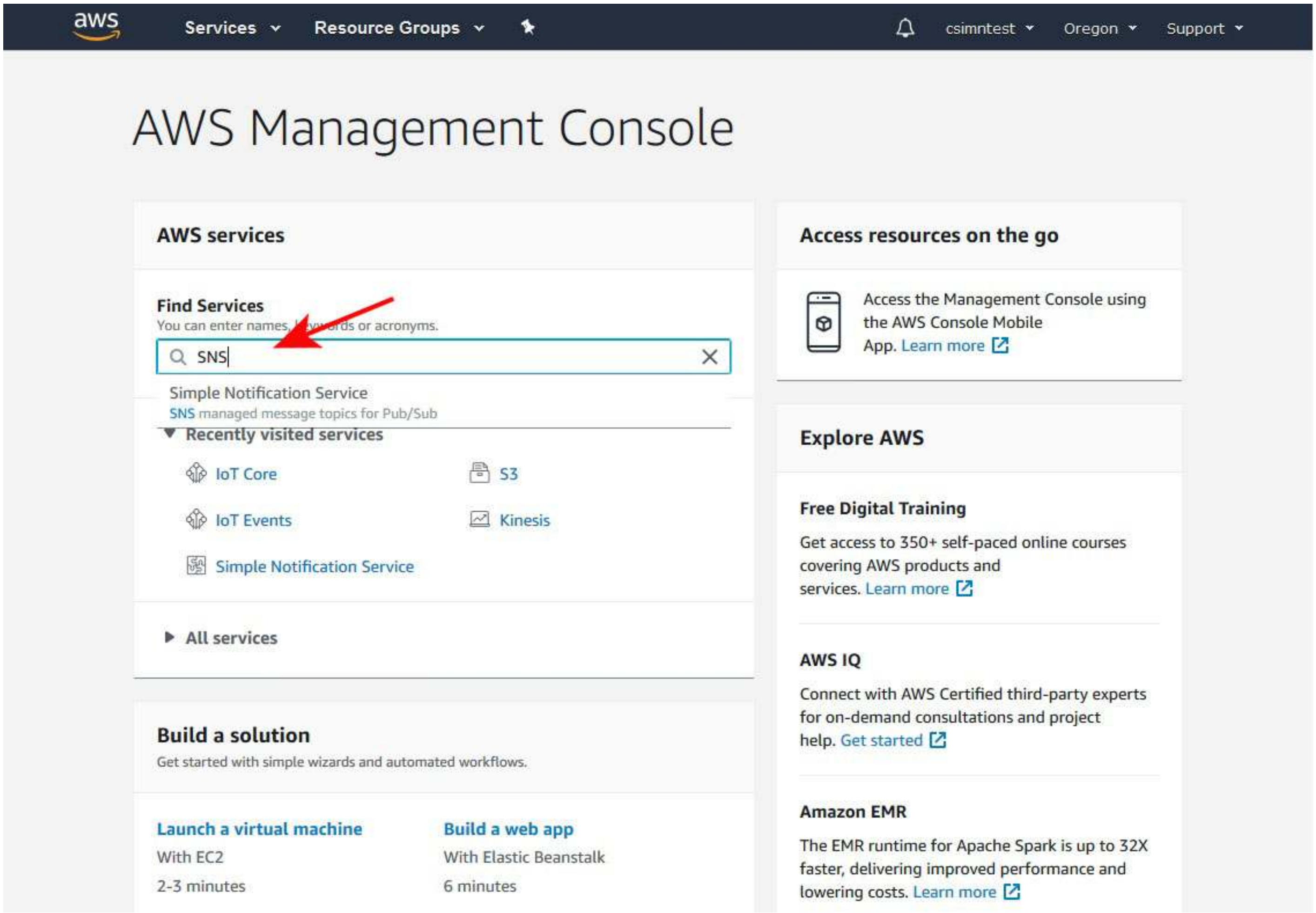
Local Register #	Register Name	Set	Register Data	Register Format
00001	csiSensor1	□	55	Signed 16-bit
00002	csiSensor2	□	28	Signed 16-bit
00003	csiSensor3	□	0	Signed 16-bit
00004	csiSensor4	□	0	Signed 16-bit
00005	csiSensor5	□	0	Signed 16-bit
00006	csiSensor6	□	0	Signed 16-bit
00007	csiSensor7	□	0	Signed 16-bit
00008	csiSensor8	□	0	Signed 16-bit
00009	csiSensor9	□	0	Signed 16-bit
00010	csiSensor10	□	0	Signed 16-bit
00011	csiActuator1	□	47	Signed 16-bit

16. Configuring AWS Simple Notification Service

The AWS Simple Notification Service (SNS) can take you down multiple paths which we encourage you to explore through Amazon's online documentation. The simplest and potentially most commonly used form of notification is the text message to your mobile phone. This section will illustrate that option.

16.1 Create Topic

Start by locating SNS on your AWS Management Console.



AWS Management Console

AWS services

Find Services
You can enter names, keywords or acronyms.

SNS

Simple Notification Service
SNS managed message topics for Pub/Sub

Recently visited services

Build a solution
Get started with simple wizards and automated workflows.

Launch a virtual machine
With EC2
2-3 minutes

Build a web app
With Elastic Beanstalk
6 minutes

Access resources on the go

Access the Management Console using the AWS Console Mobile App. Learn more

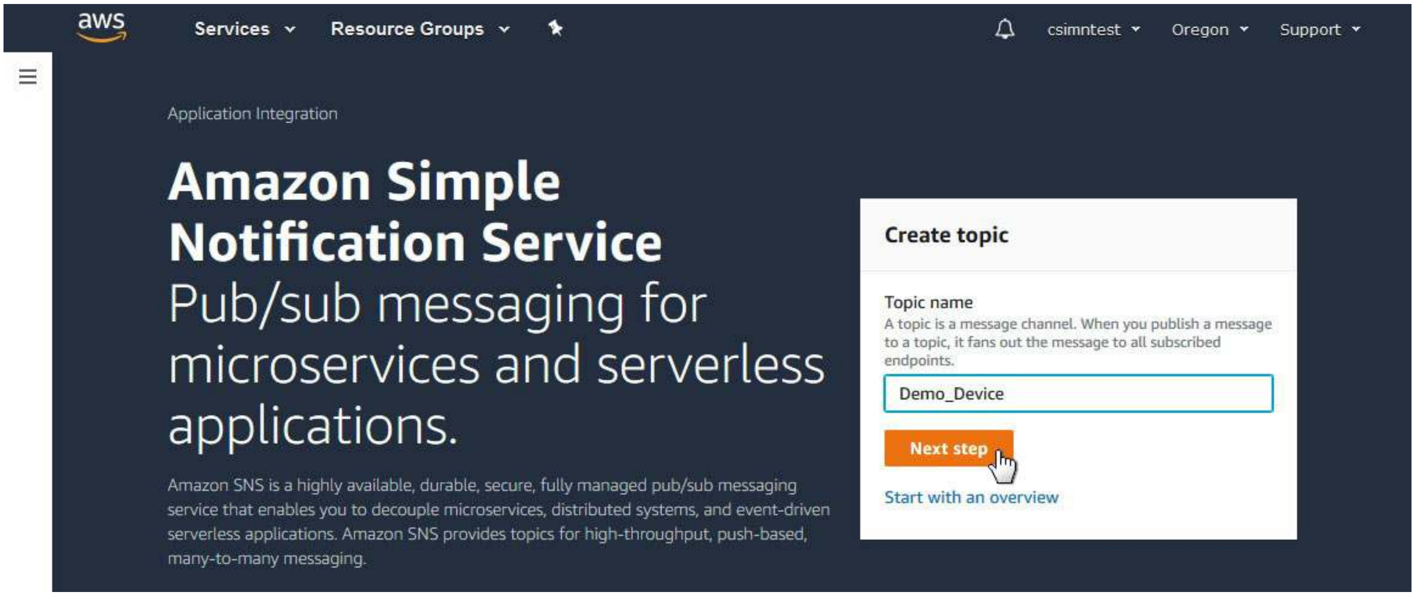
Explore AWS

Free Digital Training
Get access to 350+ self-paced online courses covering AWS products and services. Learn more

AWS IQ
Connect with AWS Certified third-party experts for on-demand consultations and project help. Get started

Amazon EMR
The EMR runtime for Apache Spark is up to 32X faster, delivering improved performance and lowering costs. Learn more

When you are starting out with no SNS topics, the screen will look something like the following. Enter a name for your SNS topic and click Next Step. Note that the SNS topic name can have no embedded spaces and no special characters other than hyphen or underscore.



Amazon Simple Notification Service
Pub/sub messaging for microservices and serverless applications.

Amazon SNS is a highly available, durable, secure, fully managed pub/sub messaging service that enables you to decouple microservices, distributed systems, and event-driven serverless applications. Amazon SNS provides topics for high-throughput, push-based, many-to-many messaging.

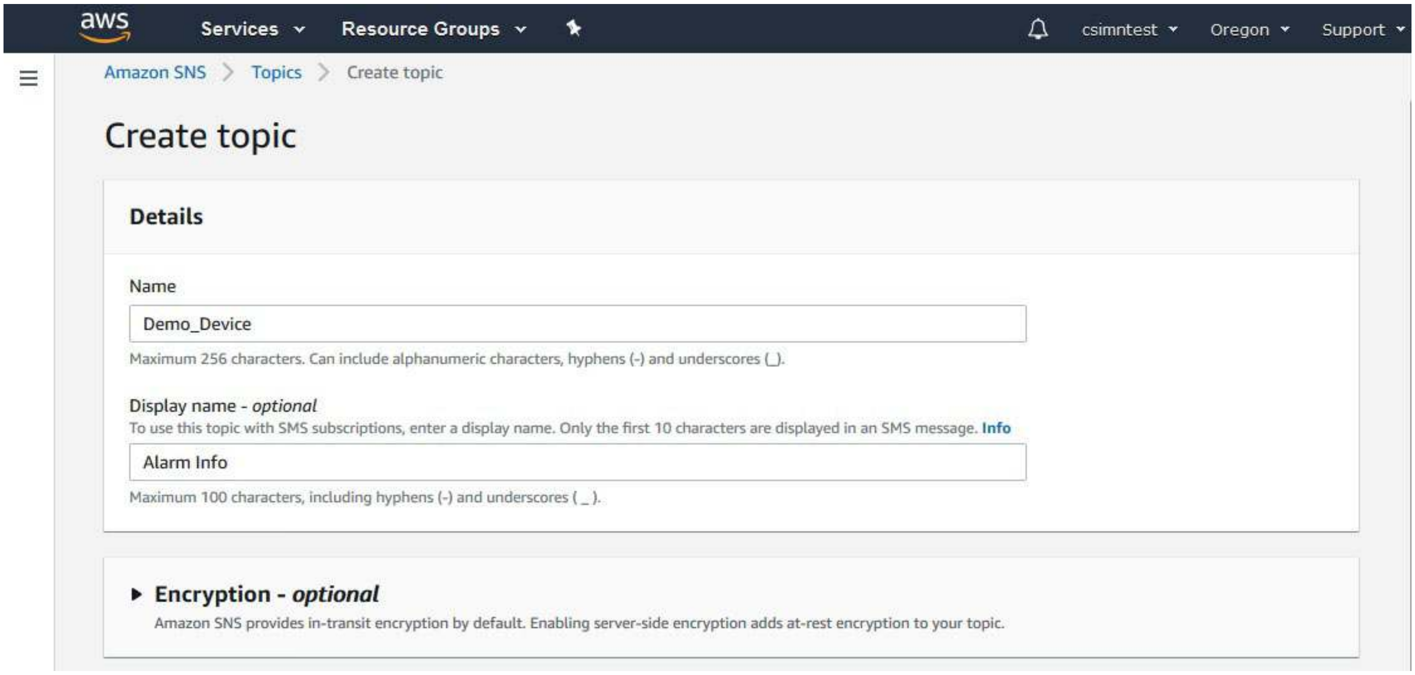
Create topic

Topic name
A topic is a message channel. When you publish a message to a topic, it fans out the message to all subscribed endpoints.

Demo_Device

You have the option of adding a display name. If provided, this is the name that will show up on your mobile phone. The display name can have spaces in it. But do note that only the first 10 characters of this display name are included in SMS messages.

You do not need any of the additional optional items that will be listed on this page.



Create topic

Details

Name

Demo_Device

Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (_).

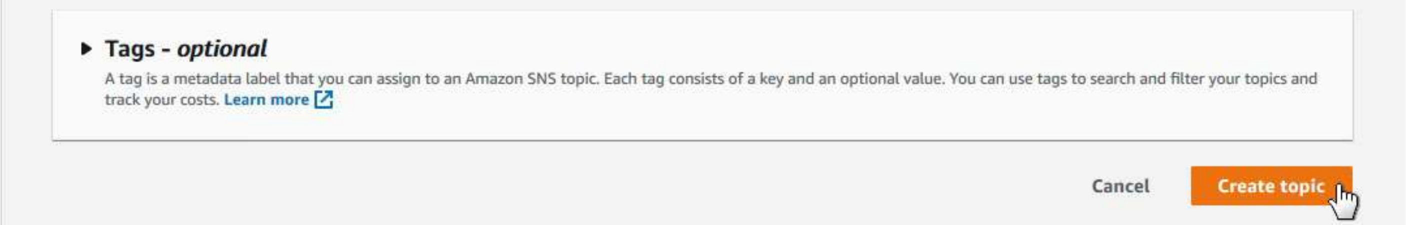
Display name - *optional*
To use this topic with SMS subscriptions, enter a display name. Only the first 10 characters are displayed in an SMS message. Info

Alarm Info

Maximum 100 characters, including hyphens (-) and underscores (_).

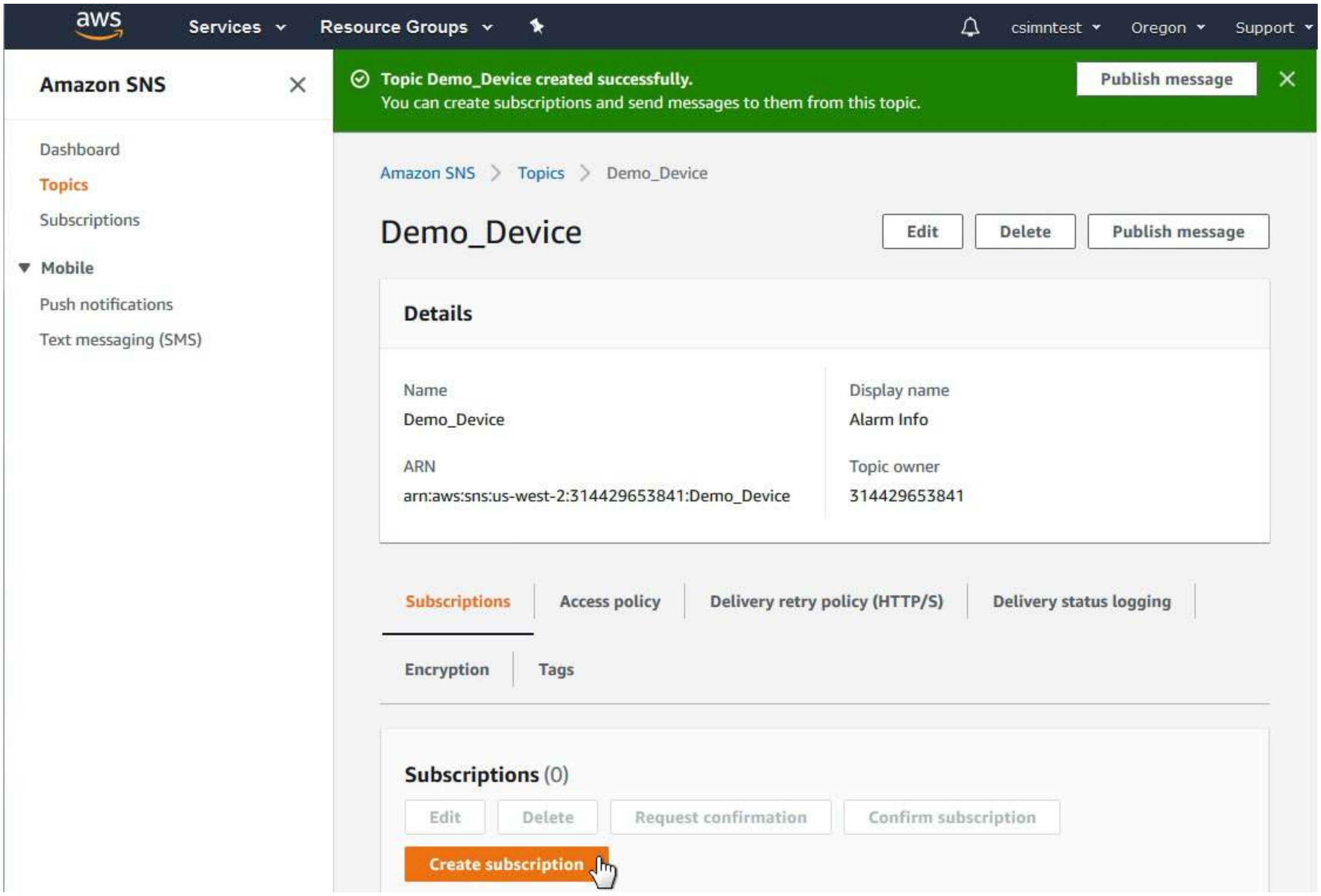
▸ Encryption - *optional*
Amazon SNS provides in-transit encryption by default. Enabling server-side encryption adds at-rest encryption to your topic.

Proceed by clicking Create Topic.


▸ Tags - *optional*
A tag is a metadata label that you can assign to an Amazon SNS topic. Each tag consists of a key and an optional value. You can use tags to search and filter your topics and track your costs. Learn more

You now have an SNS topic and it will be displayed something like this. Click on Create

Subscription next.



Topic Demo_Device created successfully.
You can create subscriptions and send messages to them from this topic.

Demo_Device

Details

Name
Demo_Device

Display name
Alarm Info

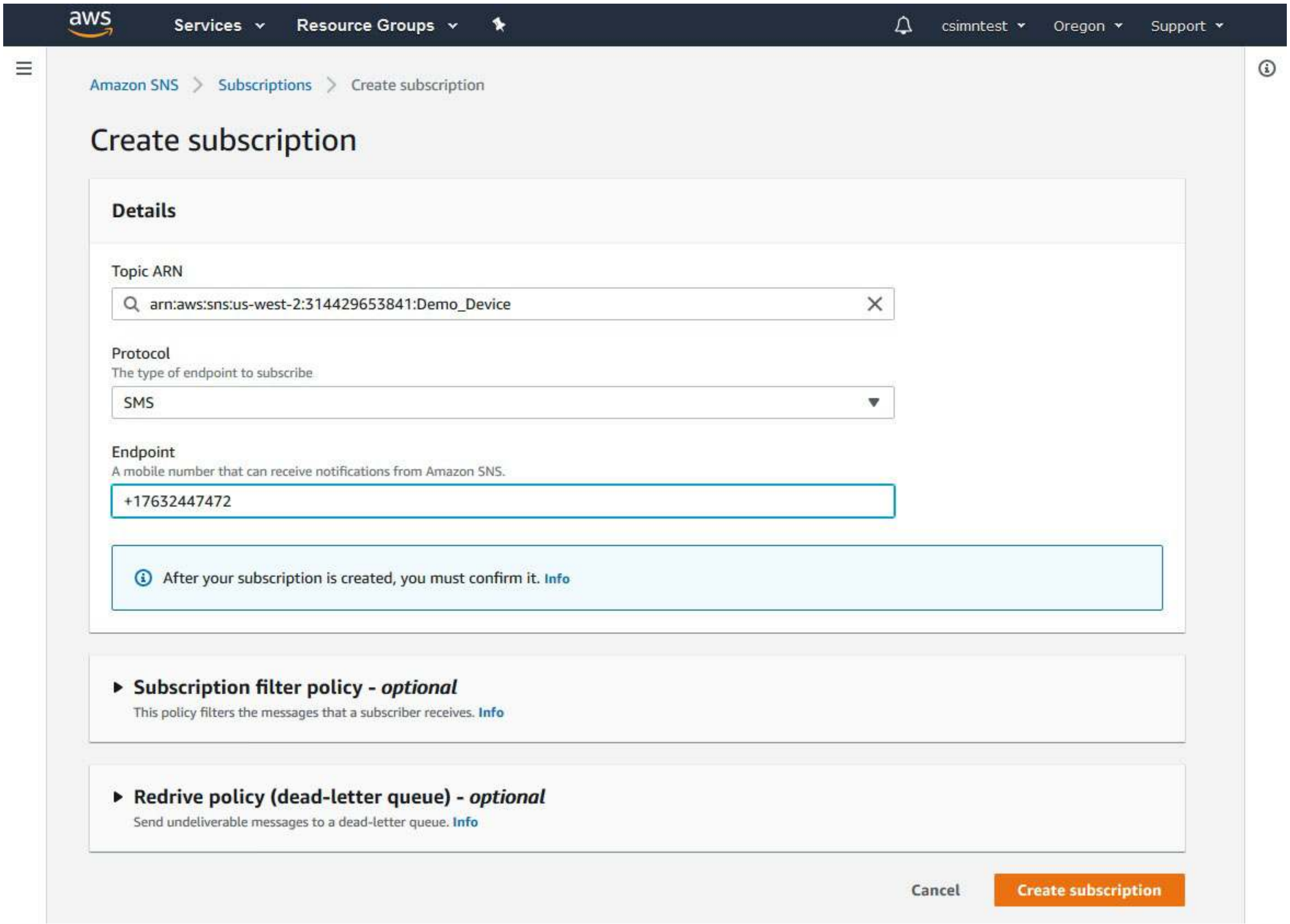
ARN
arn:aws:sns:us-west-2:314429653841:Demo_Device

Topic owner
314429653841

Subscriptions (0)

16.2 Create Subscription

The subscription to the SNS topic is where you set up the SMS part of the notification. This SMS notification will be listening for any updates to your SNS topic and send them to your phone. Select SMS for Protocol, and enter your phone number for Endpoint. Then click Create Subscription.



Create subscription

Details

Topic ARN

arn:aws:sns:us-west-2:314429653841:Demo_Device

Protocol

The type of endpoint to subscribe

SMS

Endpoint

A mobile number that can receive notifications from Amazon SNS.

+17632447472

After your subscription is created, you must confirm it. Info

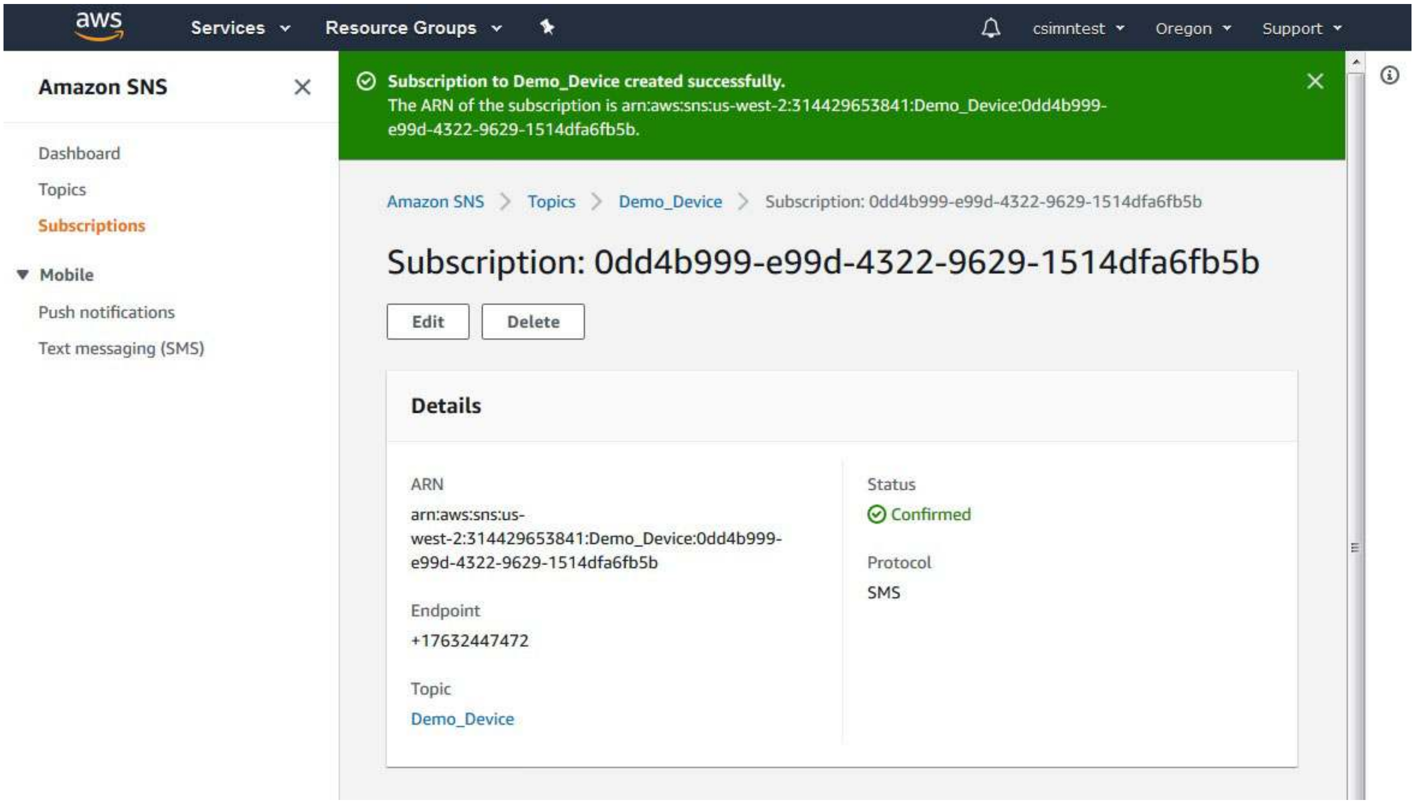
▸ Subscription filter policy - *optional*

This policy filters the messages that a subscriber receives. Info

▸ Redrive policy (dead-letter queue) - *optional*

Send undeliverable messages to a dead-letter queue. Info

You will now have an SNS subscription that looks something like this.



Subscription to Demo_Device created successfully.

The ARN of the subscription is arn:aws:sns:us-west-2:314429653841:Demo_Device:0dd4b999-e99d-4322-9629-1514dfa6fb5b.

Subscription: 0dd4b999-e99d-4322-9629-1514dfa6fb5b

Details

ARN

arn:aws:sns:us-west-2:314429653841:Demo_Device:0dd4b999-e99d-4322-9629-1514dfa6fb5b

Endpoint

+17632447472

Topic

Demo_Device

Status

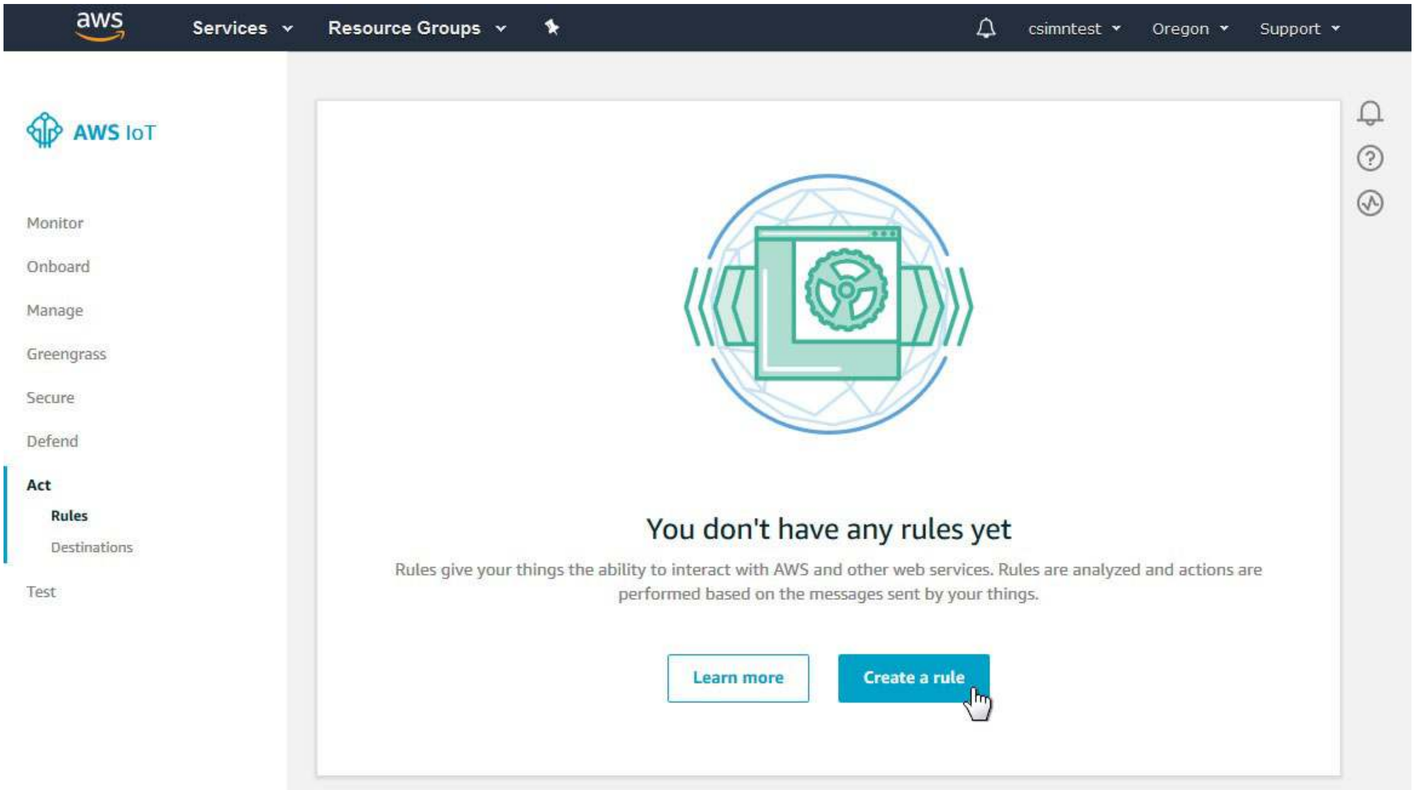
Confirmed

Protocol

SMS

16.3 Create Action Rule for SNS

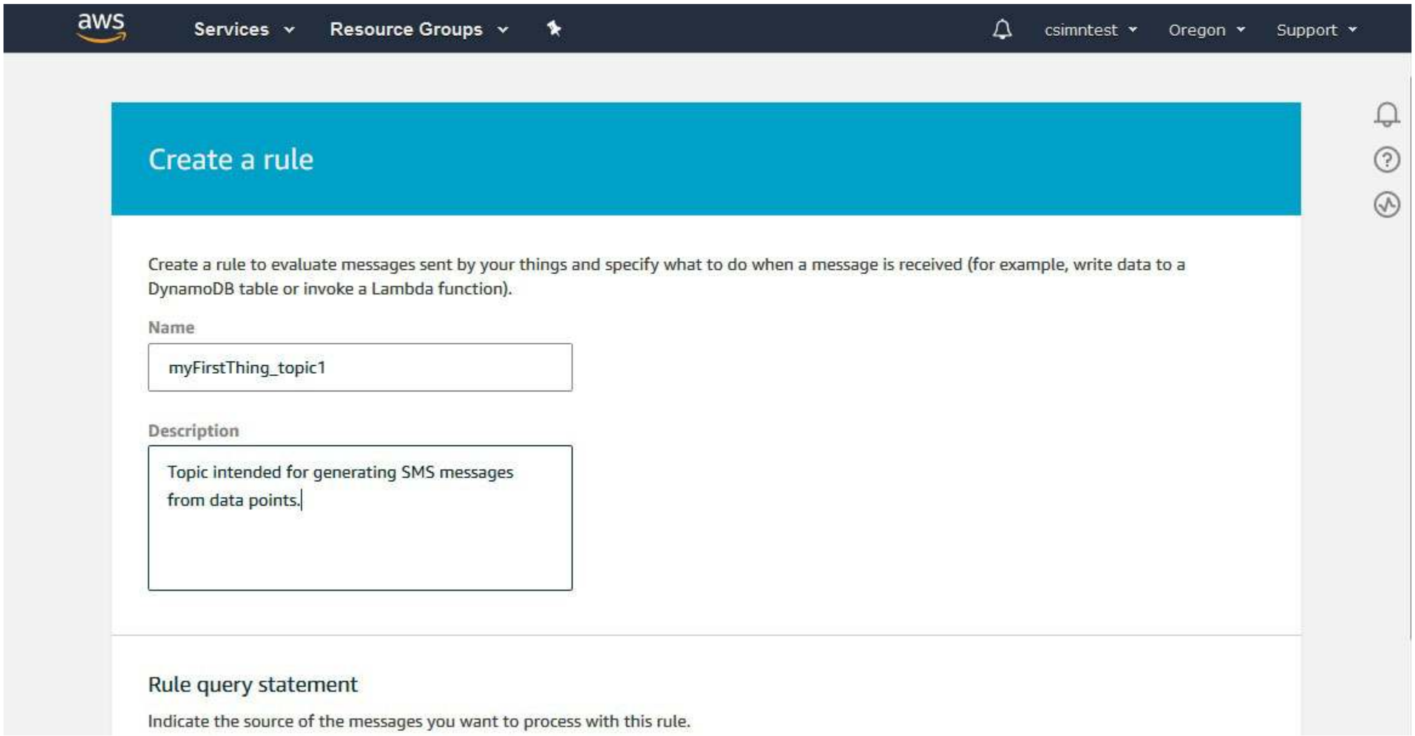
The SNS notification has now been set up, but we need to create a rule telling the system to send data to that SNS topic. Go back to IoT Core, and click on Act under the AWS IoT menu. Click on "Create a rule".



You don't have any rules yet

Rules give your things the ability to interact with AWS and other web services. Rules are analyzed and actions are performed based on the messages sent by your things.

Enter a name for your rule, and provide a description just for your own use - this doesn't get sent anywhere.



Create a rule

Create a rule to evaluate messages sent by your things and specify what to do when a message is received (for example, write data to a DynamoDB table or invoke a Lambda function).

Name

myFirstThing_topic1

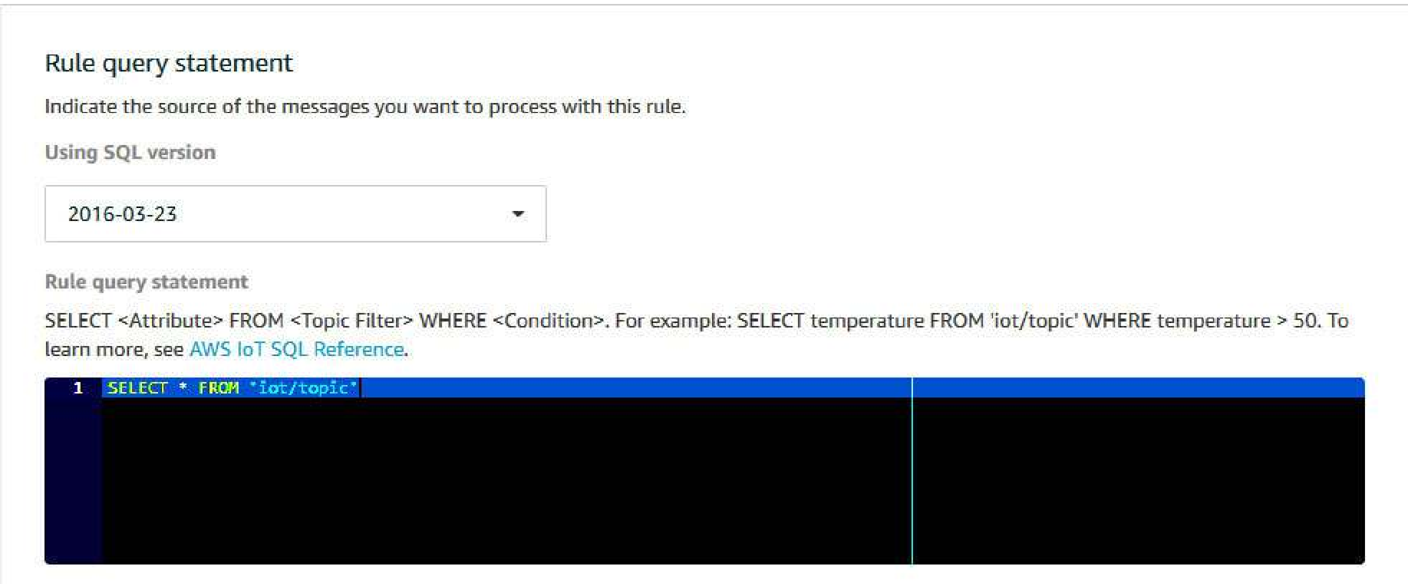
Description

Topic intended for generating SMS messages from data points.

Rule query statement

Indicate the source of the messages you want to process with this rule.

You need to provide an SQL style query telling this action rule what it should send somewhere. The default will initially just be 'iot/topic' but that is just a place holder.



Rule query statement

Indicate the source of the messages you want to process with this rule.

Using SQL version

2016-03-23

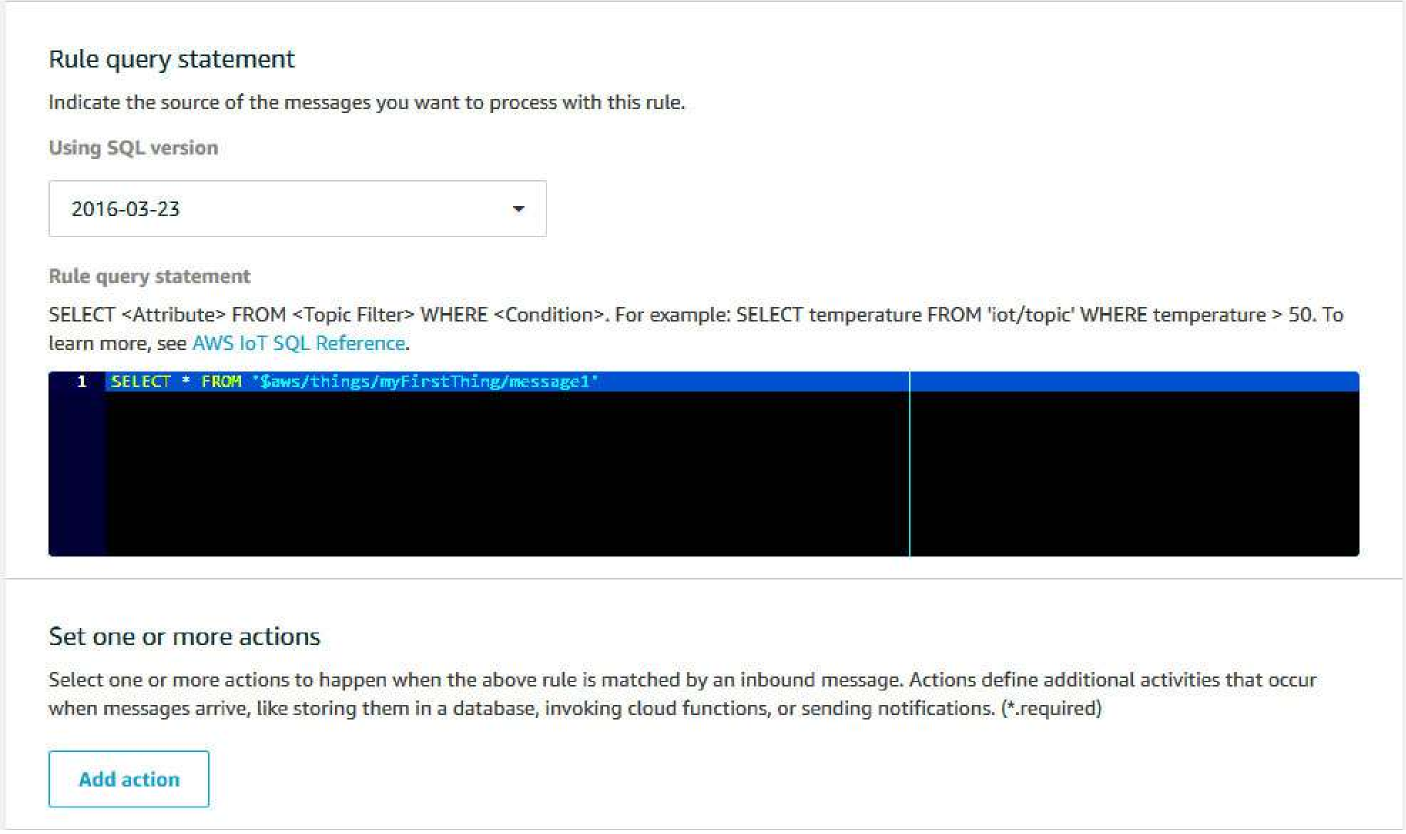
Rule query statement

SELECT <Attribute> FROM <Topic Filter> WHERE <Condition>. For example: SELECT temperature FROM 'iot/topic' WHERE temperature > 50. To learn more, see AWS IoT SQL Reference.

```
SELECT * FROM 'iot/topic'
```

The topic you want to enter here should be the exact same topic you use as the publish topic when setting up the Thing Point in the IoT Gateway. The default topic for publishing to the Shadow is '$aws/things/myFirstThing/shadow/update'. But to avoid having every data point sent to our phone, and have only specifically selected data points sent to our phone, we are going to create our own special topic for this example.

The special topic we are going to use is '$aws/things/myFirstThing/message1'.



Rule query statement

Indicate the source of the messages you want to process with this rule.

Using SQL version

2016-03-23

Rule query statement

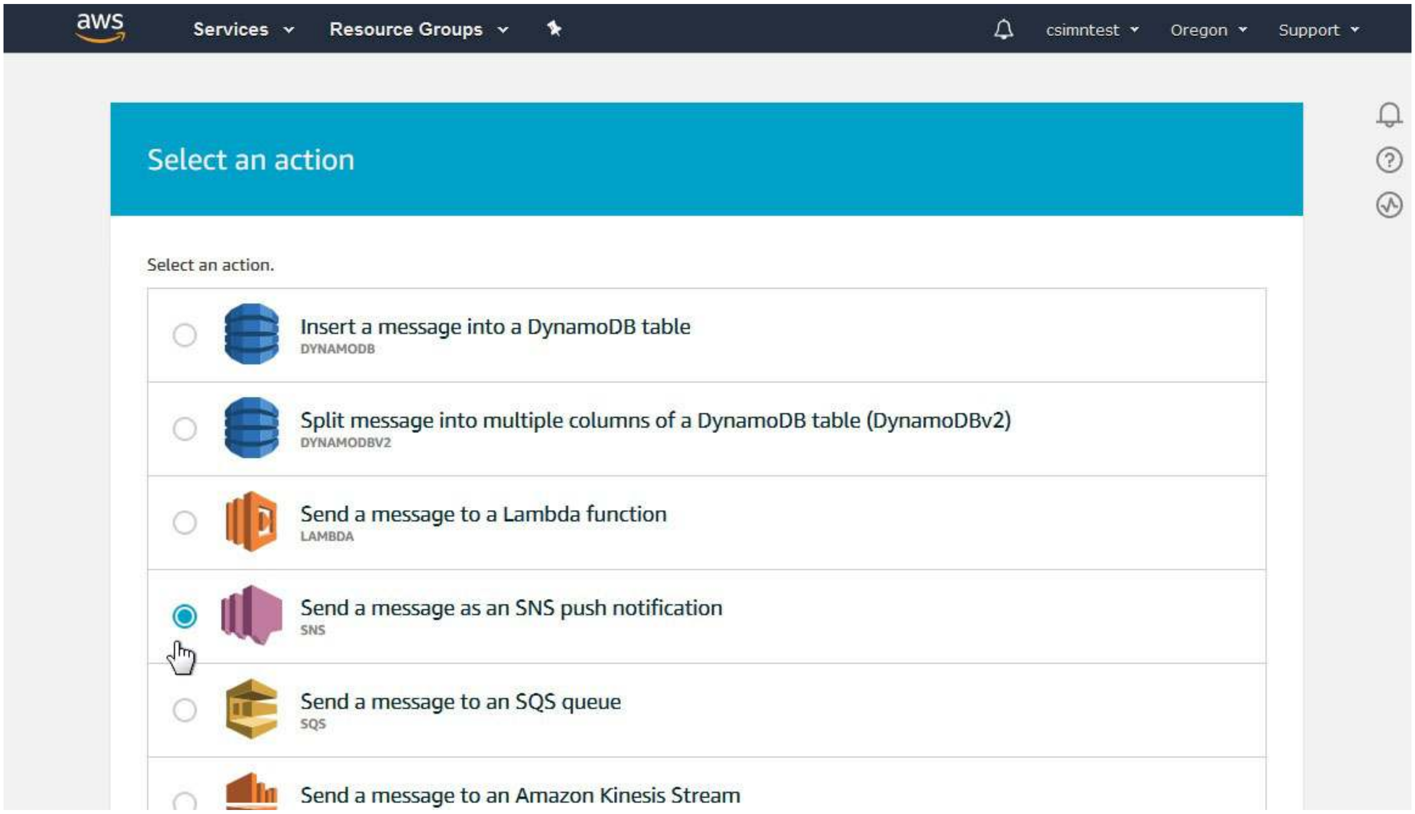
SELECT <Attribute> FROM <Topic Filter> WHERE <Condition>. For example: SELECT temperature FROM 'iot/topic' WHERE temperature > 50. To learn more, see AWS IoT SQL Reference.

```
SELECT * FROM '$aws/things/myFirstThing/message1'
```

Set one or more actions

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. (*.required)

Next, you need to add an action. Click the Add action just below the SQL query. When you do, the following screen appears. Select "Send a message as an SNS push notification".



Select an action

Select an action.

○ Insert a message into a DynamoDB table
DYNAMODB

○ Split message into multiple columns of a DynamoDB table (DynamoDBv2)
DYNAMODBV2

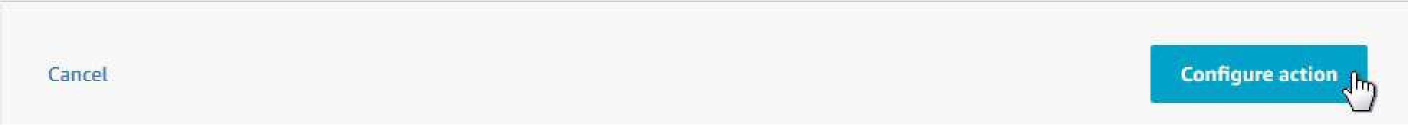
○ Send a message to a Lambda function
LAMBDA

● Send a message as an SNS push notification
SNS

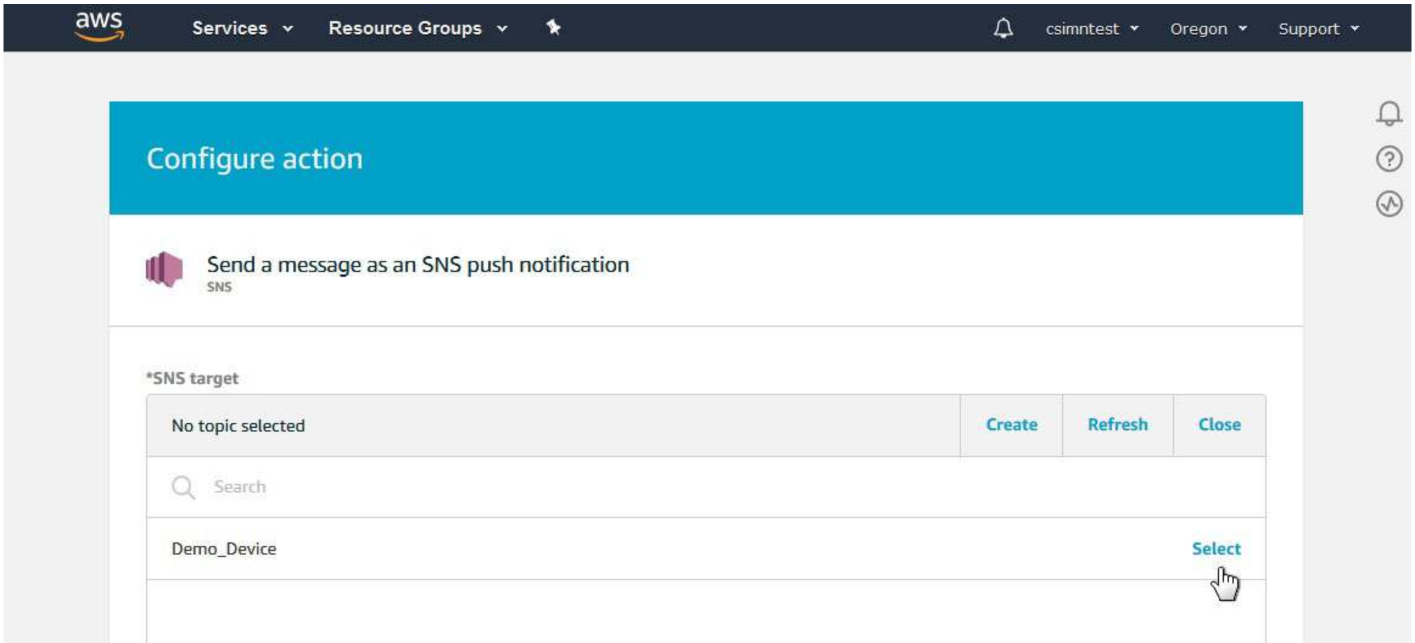
○ Send a message to an SQS queue
SQS

○ Send a message to an Amazon Kinesis Stream

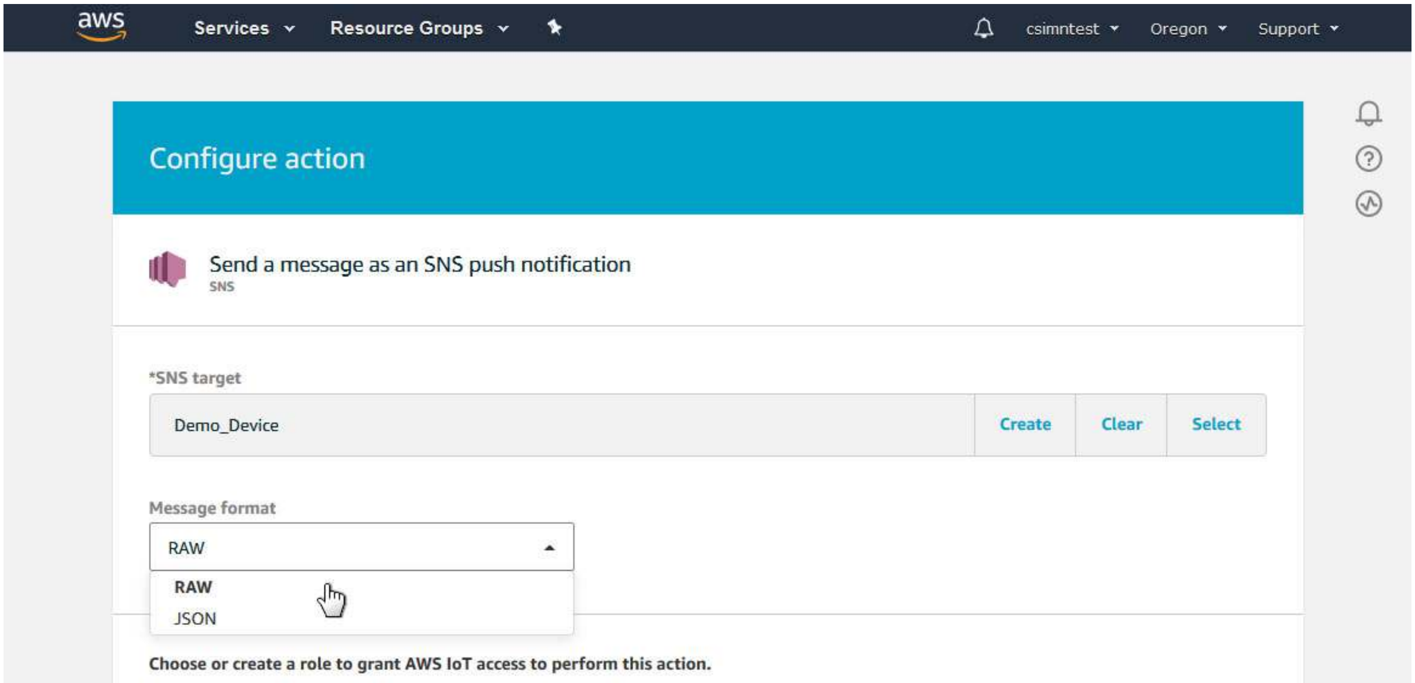
Make the SNS selection and then click Configure action.



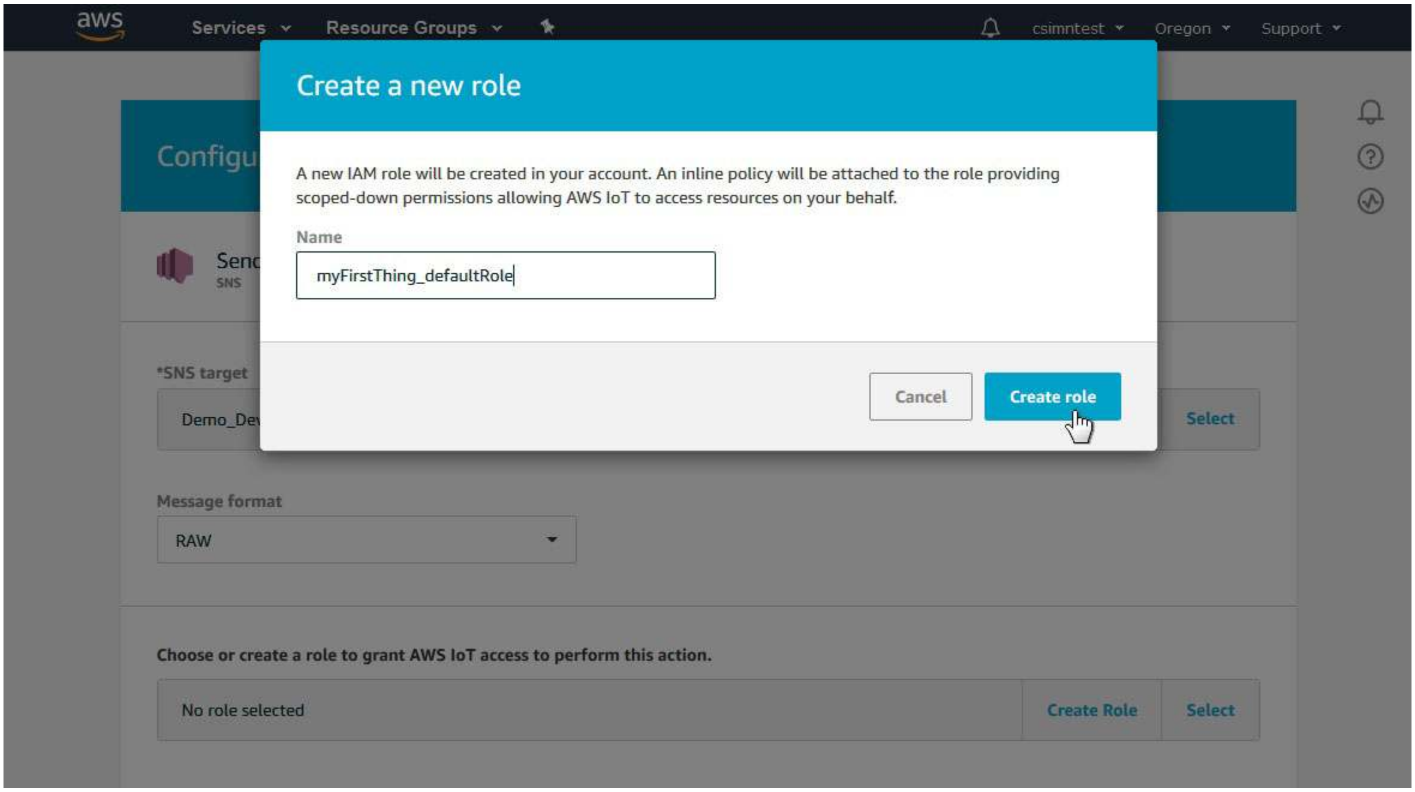
Upon clicking Configure action, the following screen appears. Here is where you will select the SNS topic created above and apply it to this Action rule. Click the Select that appears on the same line as your SNS topic.



When you select the SNS target, you also need to select Message format. You must use RAW here.



You will also see "No role selected" on this page. Click on Create Role. The window illustrated below pops up. Enter a role name, and then click Create role.



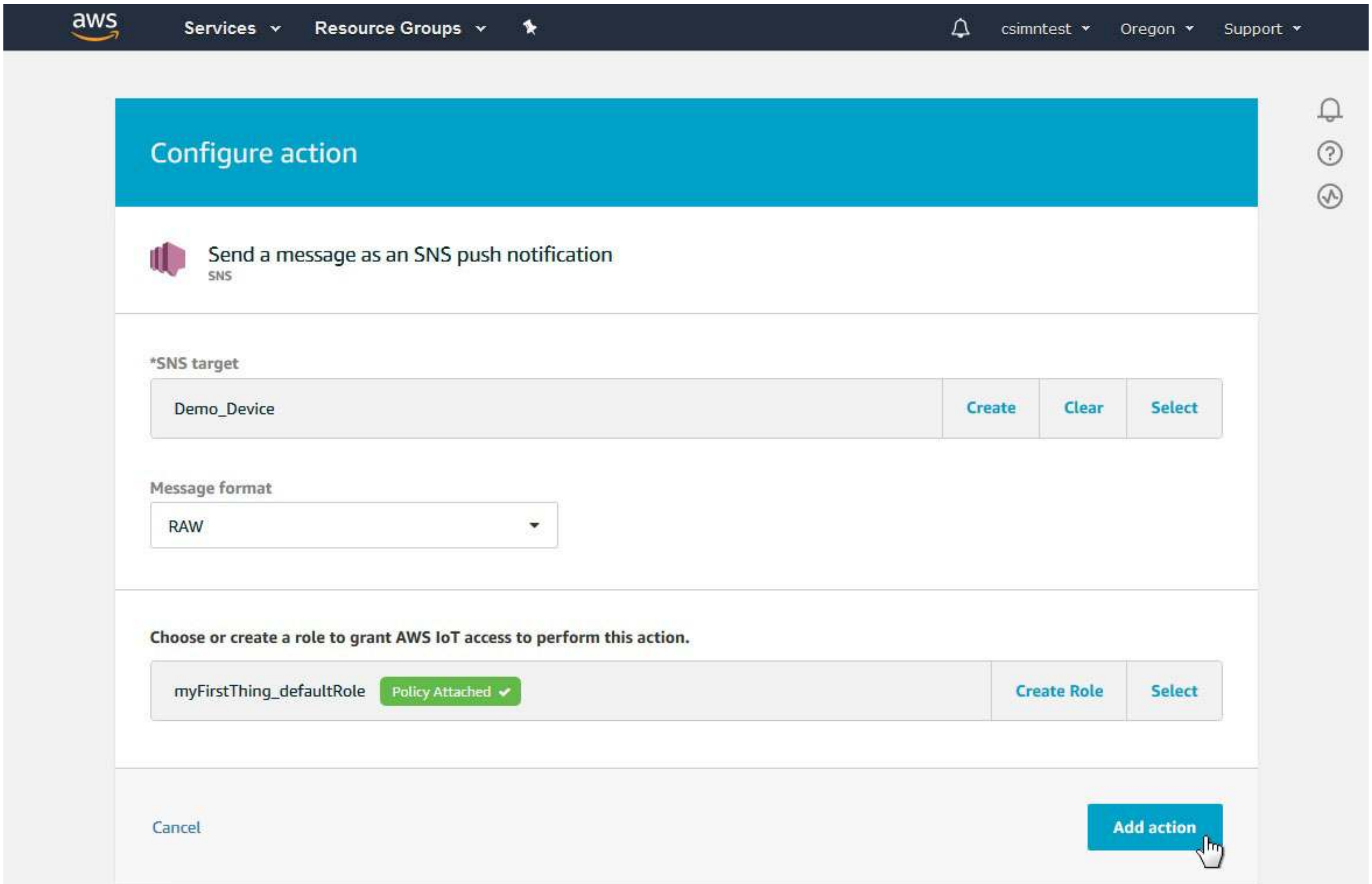
Create a new role

A new IAM role will be created in your account. An inline policy will be attached to the role providing scoped-down permissions allowing AWS IoT to access resources on your behalf.

Name: myFirstThing_defaultRole

Cancel | Create role

Your SNS push action is now configured. Click Add action.



Configure action

Send a message as an SNS push notification
SNS

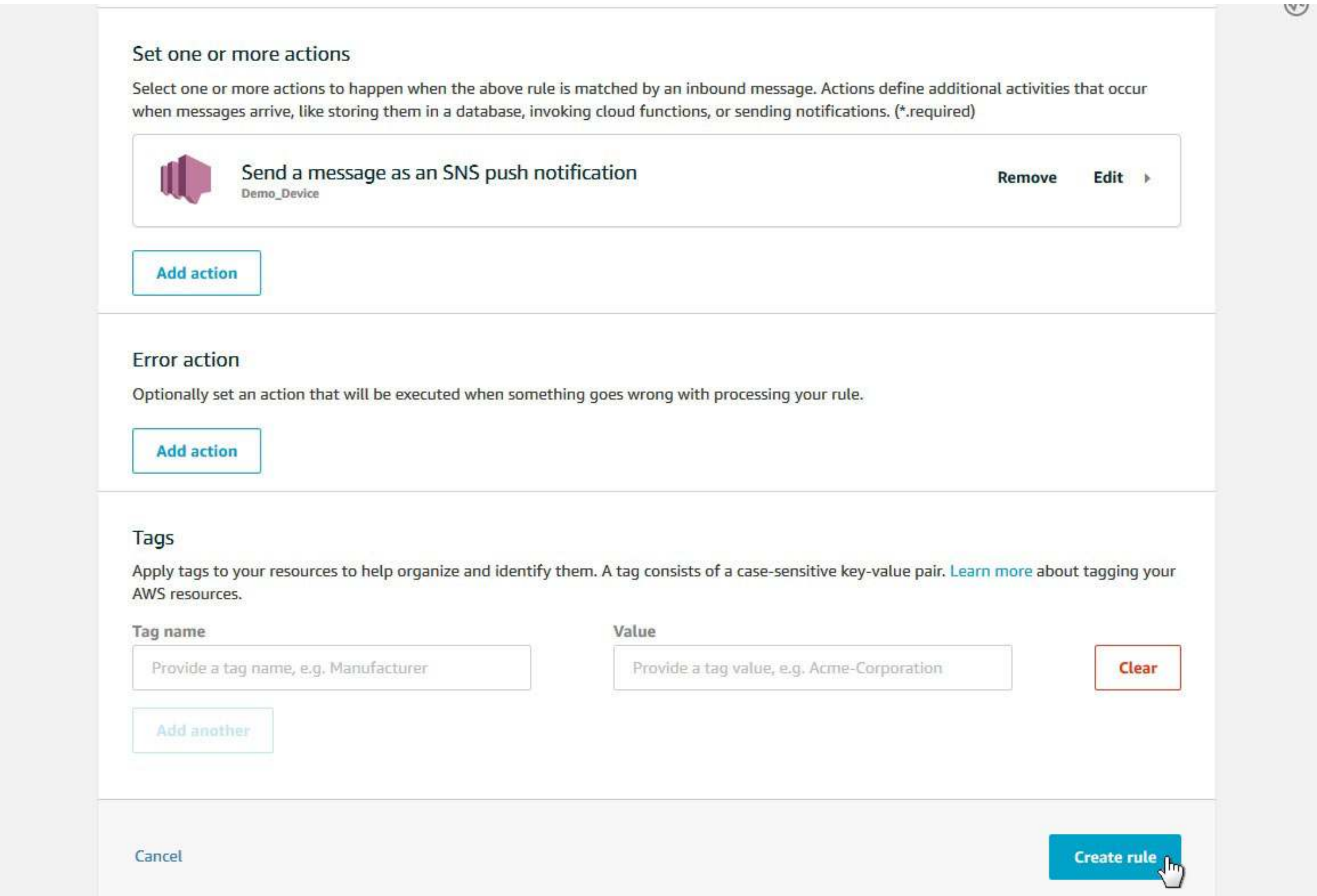
*SNS target: Demo_Device

Message format: RAW

Choose or create a role to grant AWS IoT access to perform this action.

myFirstThing_defaultRole Policy Attached ✓

Upon adding the action, the SNS action will now appear as part of the Action rule. Click on Create rule.



Set one or more actions

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. (*.required)

Send a message as an SNS push notification
Demo_Device

Error action

Optionally set an action that will be executed when something goes wrong with processing your rule.

Tags

Apply tags to your resources to help organize and identify them. A tag consists of a case-sensitive key-value pair. Learn more about tagging your AWS resources.

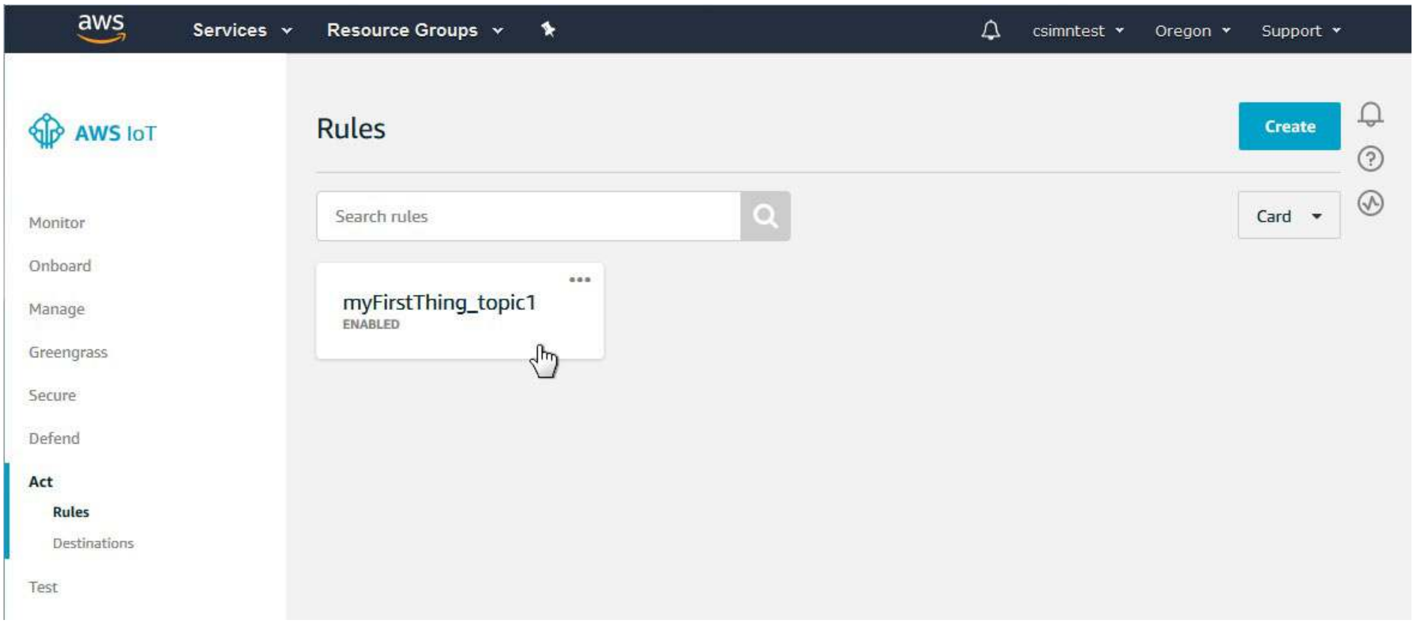
Tag name

Provide a tag name, e.g. Manufacturer

Value

Provide a tag value, e.g. Acme-Corporation

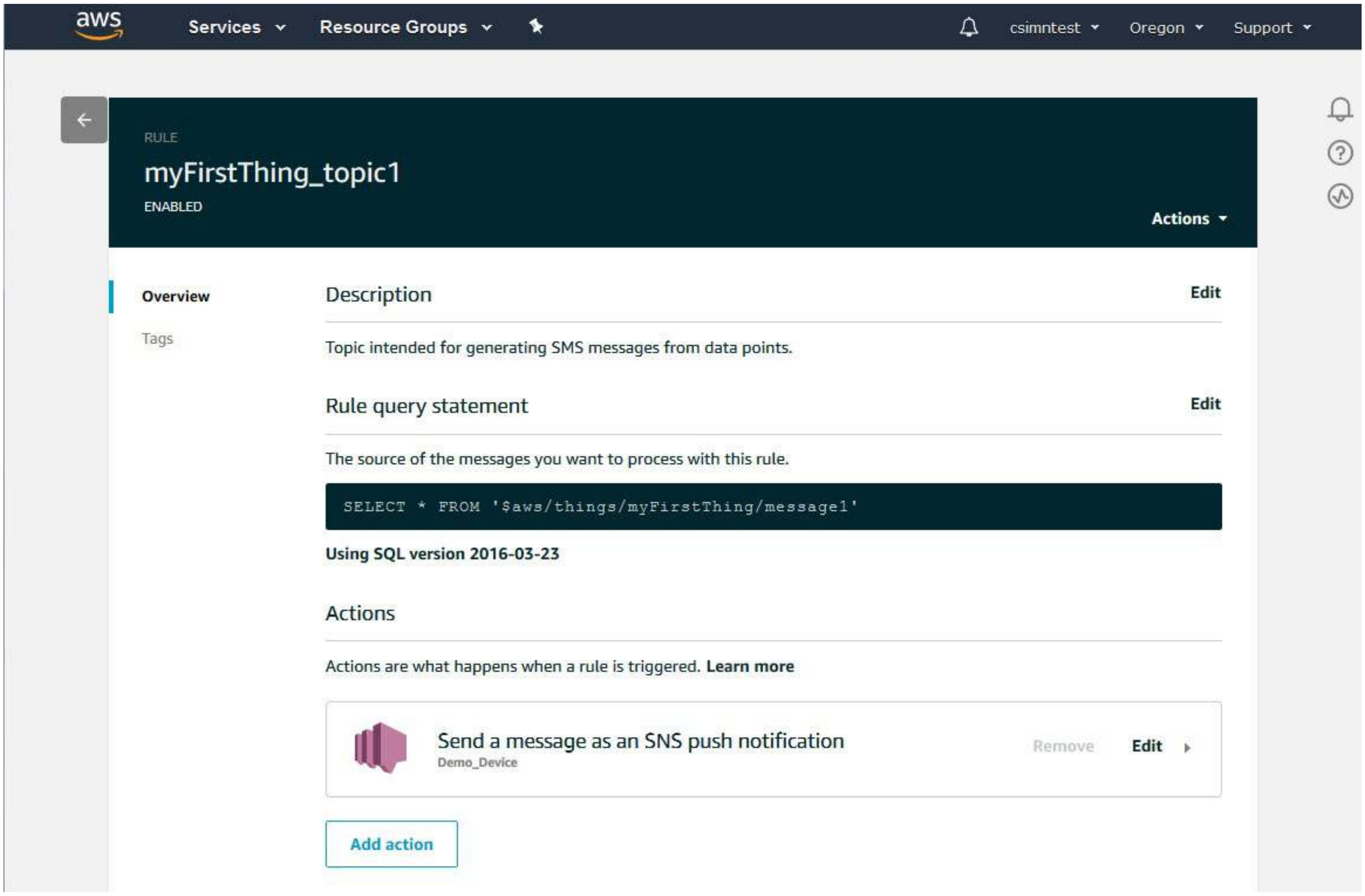
The screen that originally said "you don't have any rules yet" will now look like this:



Rules

myFirstThing_topic1
ENABLED

If you click on the rule, the configured action rule looks like this:



RULE

myFirstThing_topic1

ENABLED

Description

Topic intended for generating SMS messages from data points.

Rule query statement

The source of the messages you want to process with this rule.

```
SELECT * FROM '$aws/things/myFirstThing/message1'
```

Using SQL version 2016-03-23

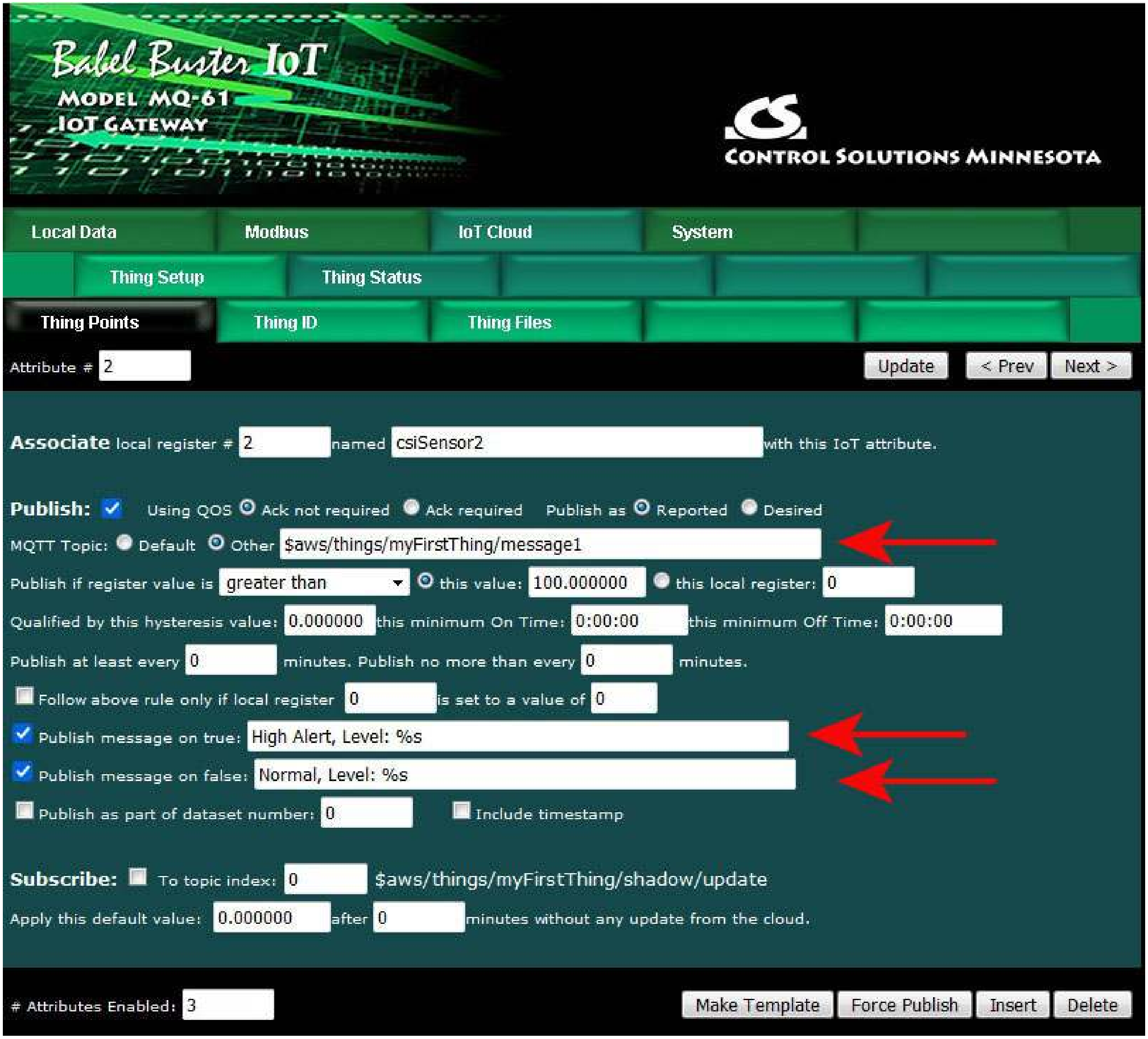
Actions

Actions are what happens when a rule is triggered. Learn more

Send a message as an SNS push notification
Demo_Device

16.4 Configure IoT Gateway and Send Text Message

Most of the work is done. The only thing that remains is to configure a Thing Point to send a message to the SNS topic. Notice the special topic we created above is used here as the "Other" topic for publishing. Also note that we filled in messages that will be of more interest than just a number.



Attribute # 2

Associate local register # 2 named csiSensor2 with this IoT attribute.

Publish: ☑ Using QOS ● Ack not required ○ Ack required Publish as ● Reported ○ Desired

MQTT Topic: ○ Default ● Other $aws/things/myFirstThing/message1

Publish if register value is greater than ● this value: 100.000000 ○ 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: High Alert, Level: %s

☑ Publish message on false: Normal, Level: %s

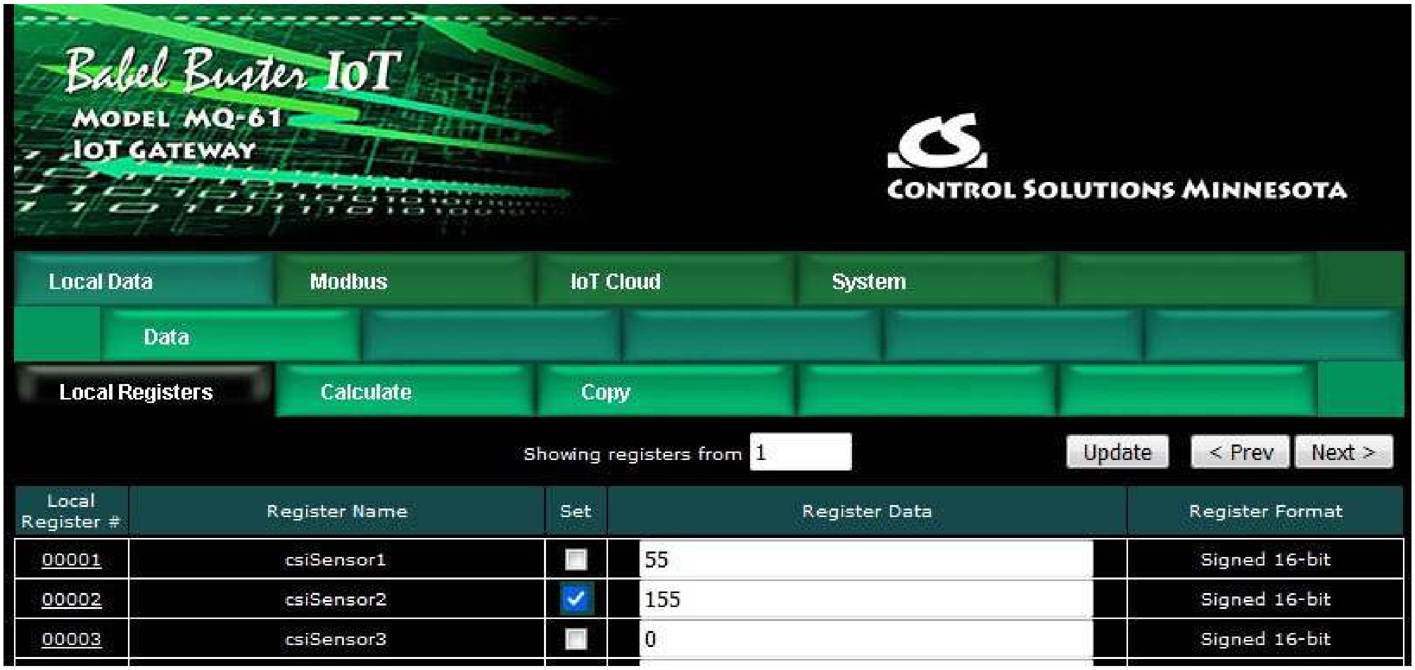
□ Publish as part of dataset number: 0 □ Include timestamp

Subscribe: □ To topic index: 0 $aws/things/myFirstThing/shadow/update

Apply this default value: 0.000000 after 0 minutes without any update from the cloud.

Attributes Enabled: 3

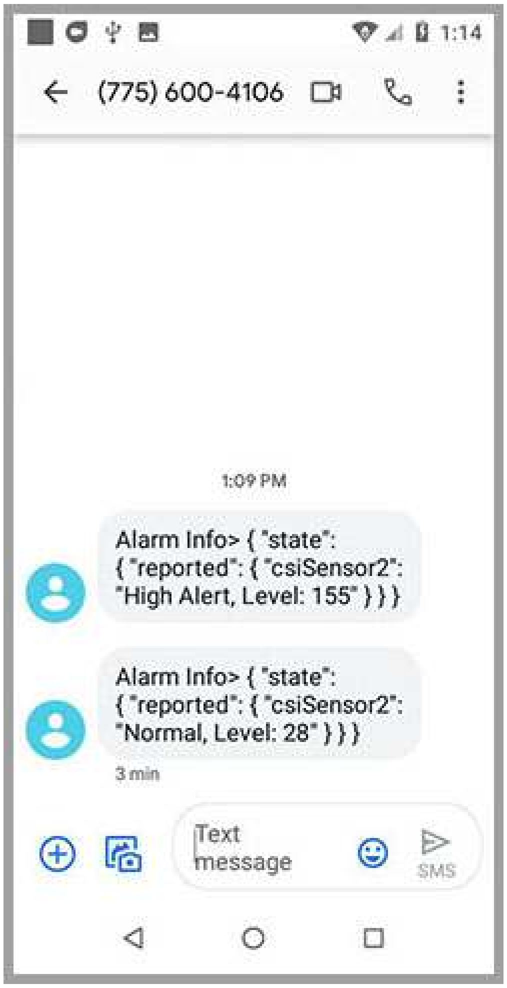
To cause the rule to publish, enter a value that triggers the rule.



Showing registers from 1

Local Register #	Register Name	Set	Register Data	Register Format
00001	csiSensor1	☐	55	Signed 16-bit
00002	csiSensor2	☑	155	Signed 16-bit
00003	csiSensor3	☐	0	Signed 16-bit

The message that appeared on the cell phone in this test is illustrated below. In this case, the rule was first triggered with the high level. Then a second level was entered that was below the threshold.



1:09 PM

Alarm Info> { "state": { "reported": { "csiSensor2": "High Alert, Level: 155" } } }

Alarm Info> { "state": { "reported": { "csiSensor2": "Normal, Level: 28" } } }

3 min

Babel Buster IoT
MODEL MQ-61
MODEL BB3-6101-MQ
IOT GATEWAY

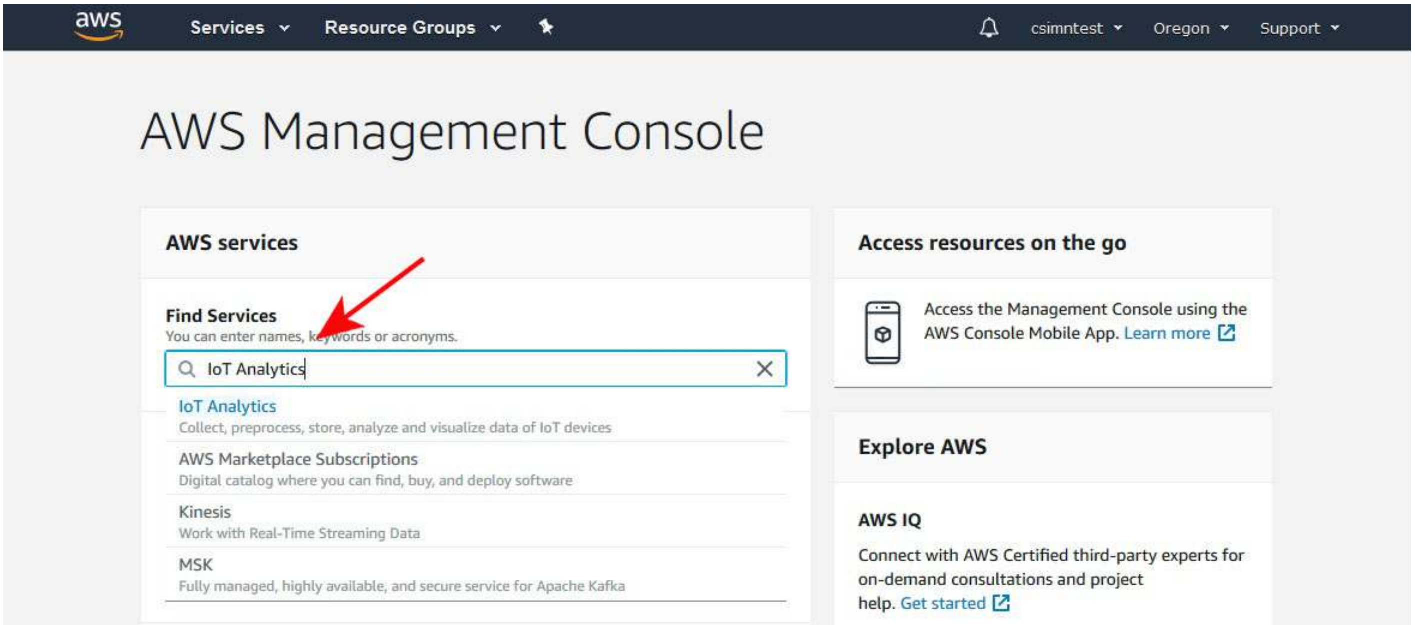
CONTROL SOLUTIONS MINNESOTA

17. Configuring IoT Analytics

Just as the subtitle denotes, IoT Analytics is where you collect, preprocess, store, analyze and visualize data of IoT devices. In this section of the user guide, we will go through the steps of setting up the channel, pipeline, data store and data set for collecting data from your Thing.

17.1 Create Resources

Start by searching for IoT Analytics and going there.



AWS Management Console

AWS services

Find Services
You can enter names, keywords or acronyms.

IoT Analytics

IoT Analytics
Collect, preprocess, store, analyze and visualize data of IoT devices

AWS Marketplace Subscriptions
Digital catalog where you can find, buy, and deploy software

Kinesis
Work with Real-Time Streaming Data

MSK
Fully managed, highly available, and secure service for Apache Kafka

Access resources on the go

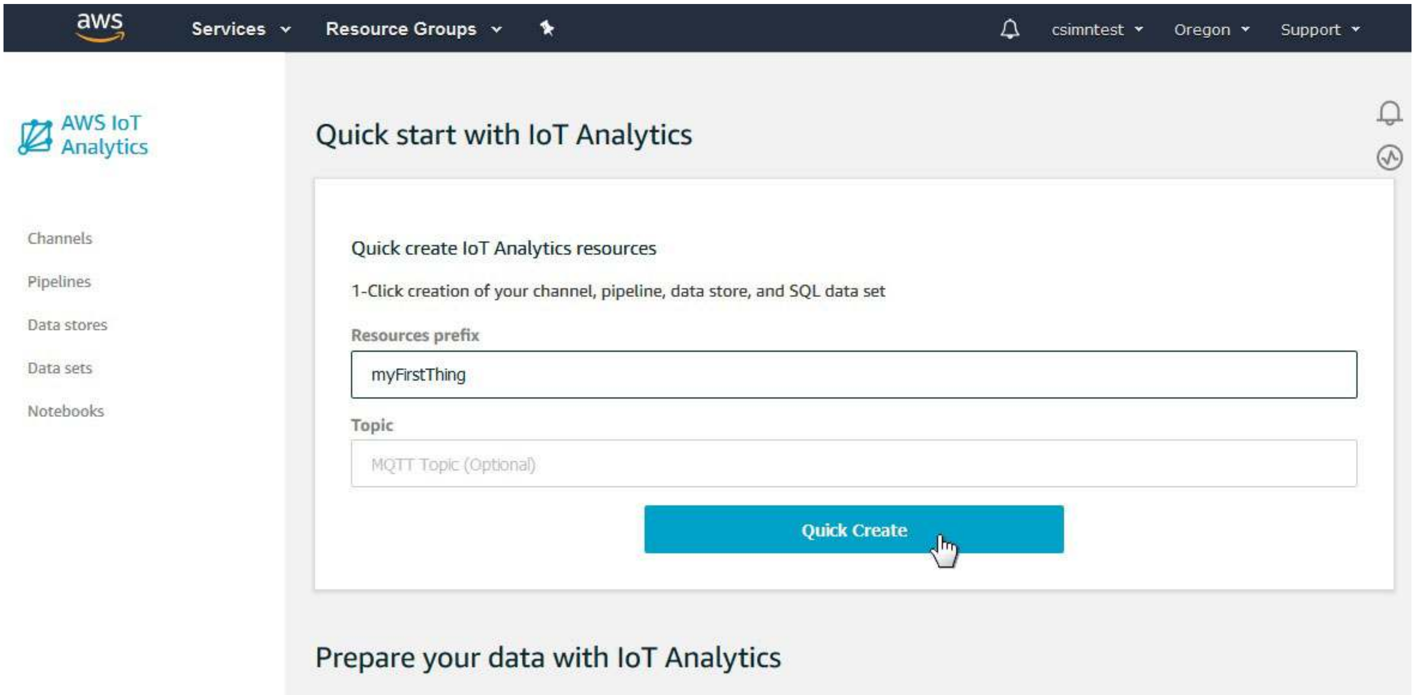
Access the Management Console using the AWS Console Mobile App. Learn more

Explore AWS

AWS IQ

Connect with AWS Certified third-party experts for on-demand consultations and project help. Get started

You may go directly to clicking on any of the menu items in the left column if you have previously set up IoT Analytics. If this is your first visit to IoT Analytics, or if you wish to set up an additional data set, enter a resource prefix and then click Quick Create.



Quick start with IoT Analytics

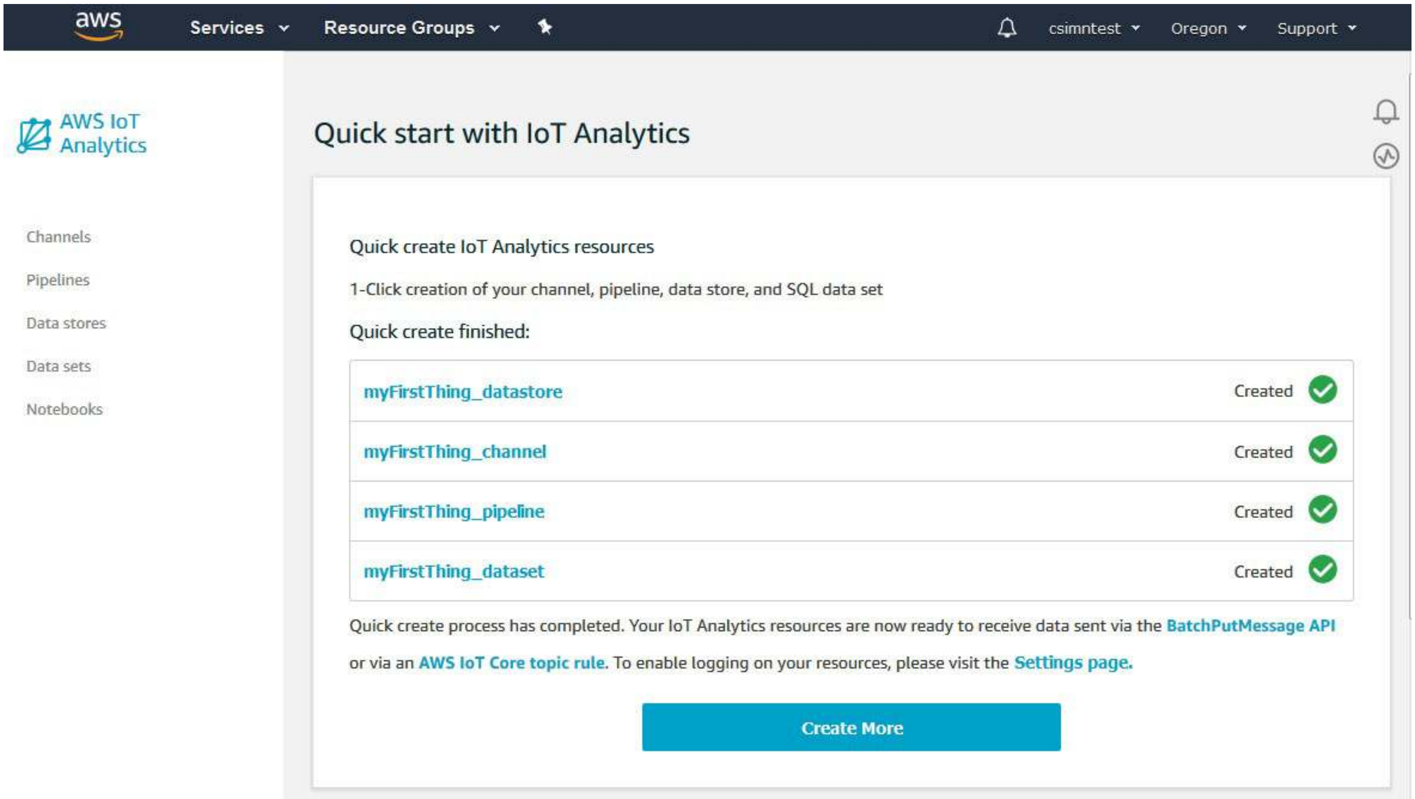
Quick create IoT Analytics resources

1-Click creation of your channel, pipeline, data store, and SQL data set

Resources prefix: myFirstThing

Topic: MQTT Topic (Optional)

The Quick Create process will create each of the four resources used in IoT Analytics. There is nothing more you need to do with the channel or data store. But you do need to configure the pipeline.



Quick start with IoT Analytics

Quick create IoT Analytics resources

1-Click creation of your channel, pipeline, data store, and SQL data set

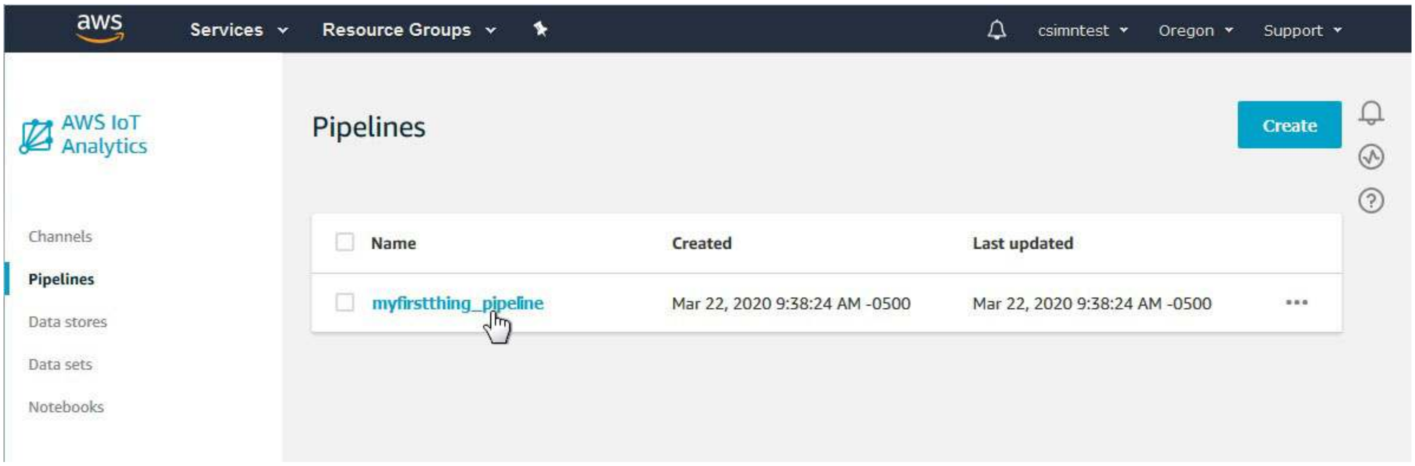
Quick create finished:

myFirstThing_datastore	Created ✓
myFirstThing_channel	Created ✓
myFirstThing_pipeline	Created ✓
myFirstThing_dataset	Created ✓

Quick create process has completed. Your IoT Analytics resources are now ready to receive data sent via the BatchPutMessage API or via an AWS IoT Core topic rule. To enable logging on your resources, please visit the Settings page.

17.2 Build the Pipeline

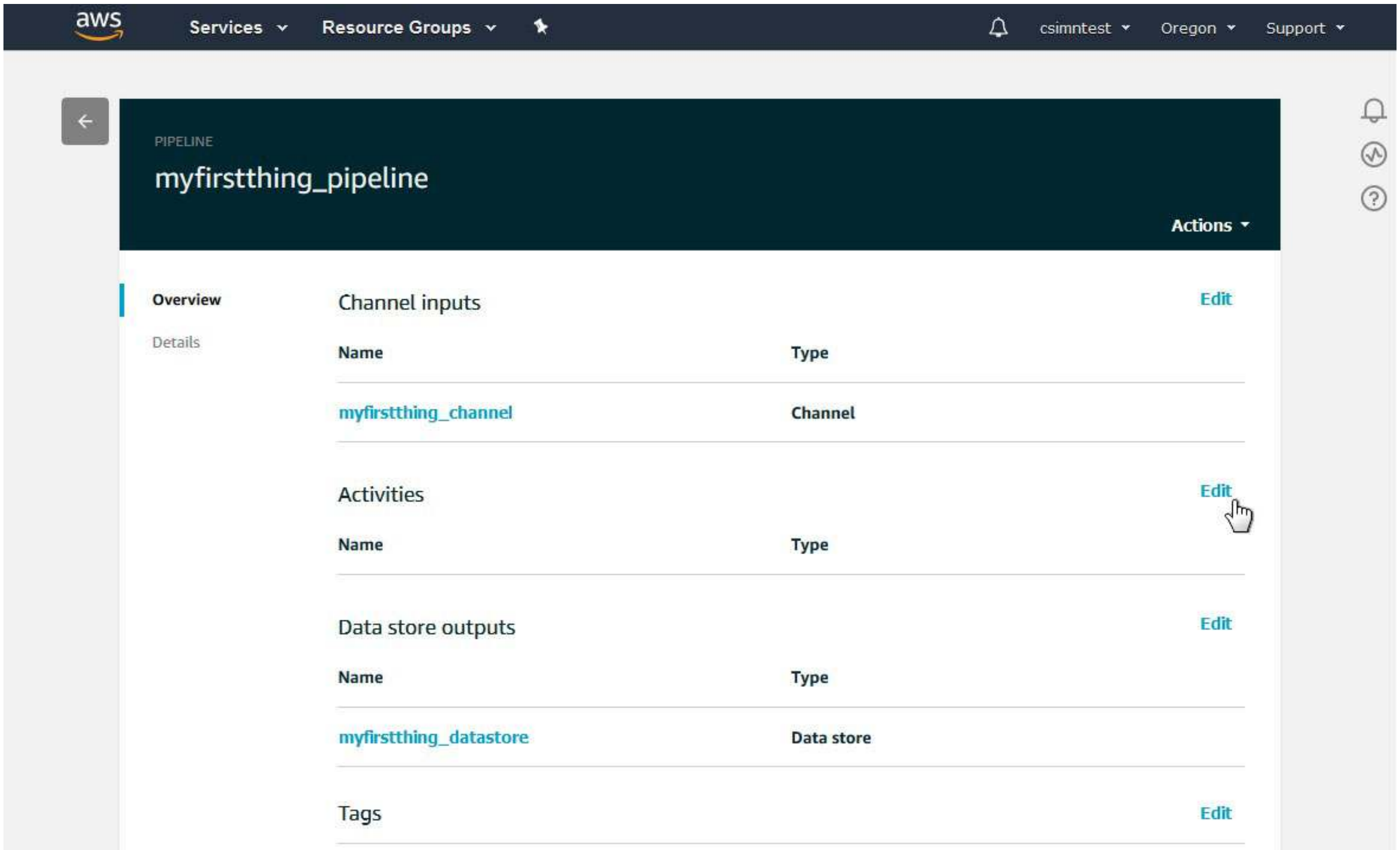
Click on Pipelines on the menu on the left, and then click on your pipeline on the list.



Pipelines

Name	Created	Last updated
myfirstthing_pipeline	Mar 22, 2020 9:38:24 AM -0500	Mar 22, 2020 9:38:24 AM -0500

You will see that your channel is already set to be the source of data, and your data store is already set to be the destination for your data. What the pipeline needs to do is process the data. So next to Activities, click Edit.



PIPELINE
myfirstthing_pipeline

Channel inputs

Name	Type
myfirstthing_channel	Channel

Activities

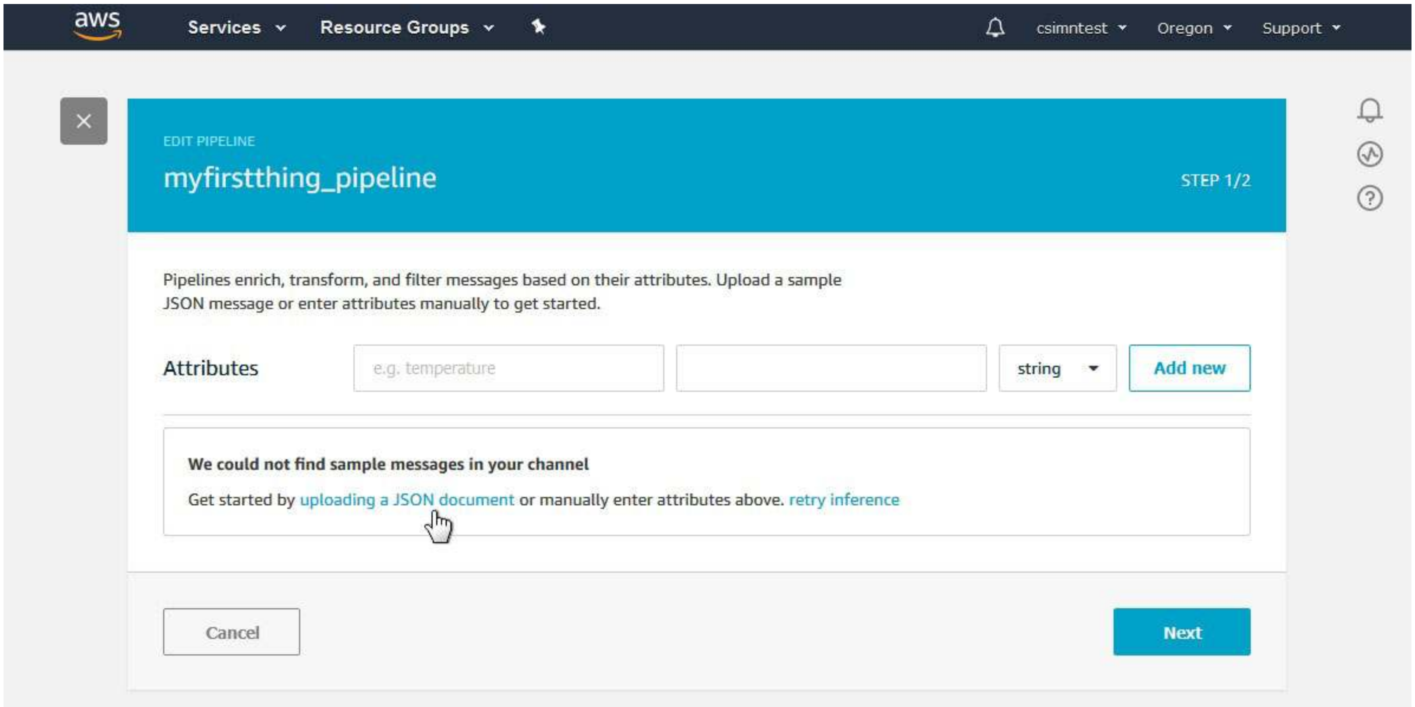
Name	Type

Data store outputs

Name	Type
myfirstthing_datastore	Data store

Tags

If you had already begun publishing data to your data channel, then the AWS system will inspect that data and make some assumptions about its format. But that requires going back and forth with partial configurations and becomes a chicken versus the egg problem. So the best and cleanest approach is to upload a JSON document telling the AWS system what your data is going to look like.



myfirstthing_pipeline

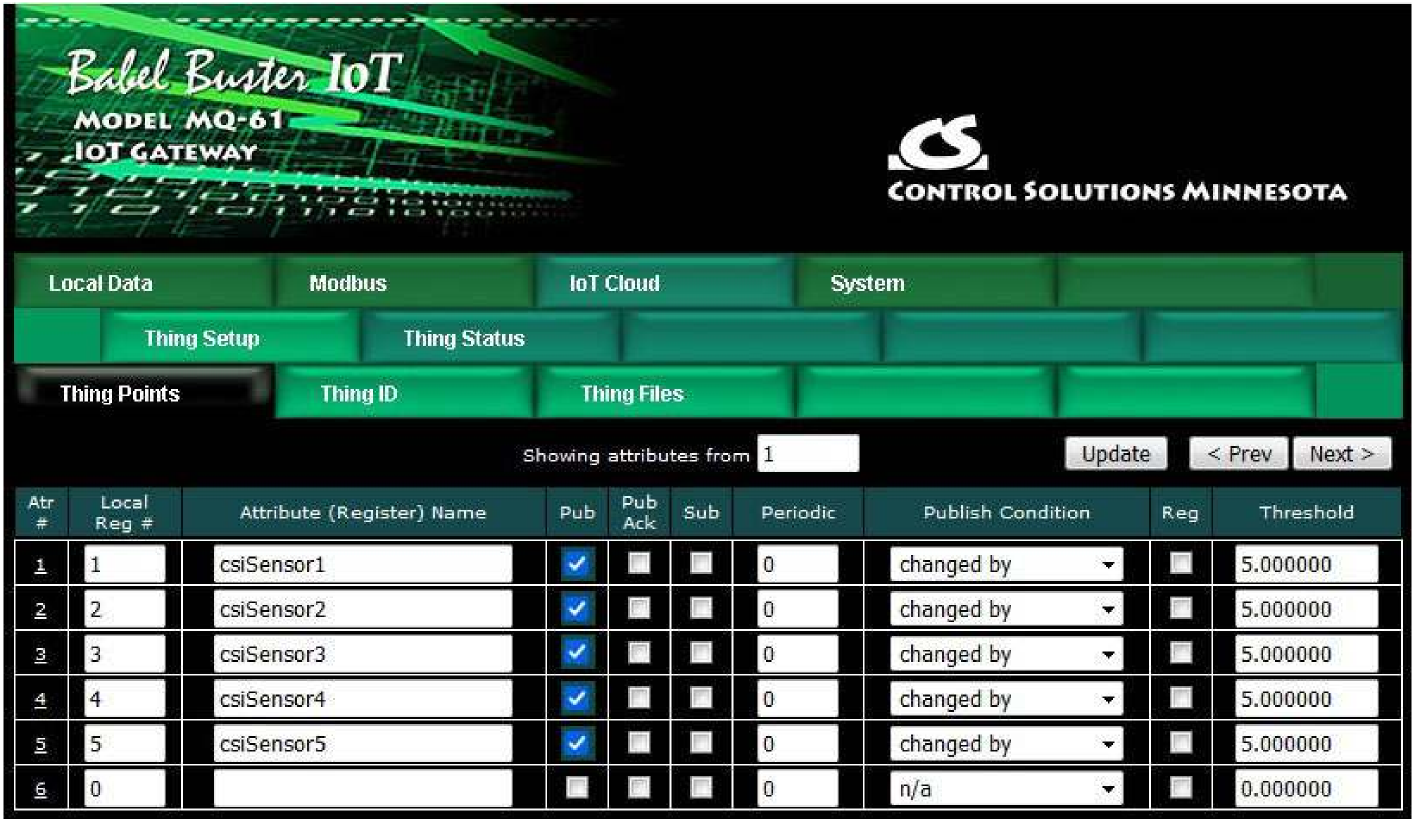
Pipelines enrich, transform, and filter messages based on their attributes. Upload a sample JSON message or enter attributes manually to get started.

Attributes: e.g. temperature

We could not find sample messages in your channel

Get started by uploading a JSON document or manually enter attributes above. retry inference

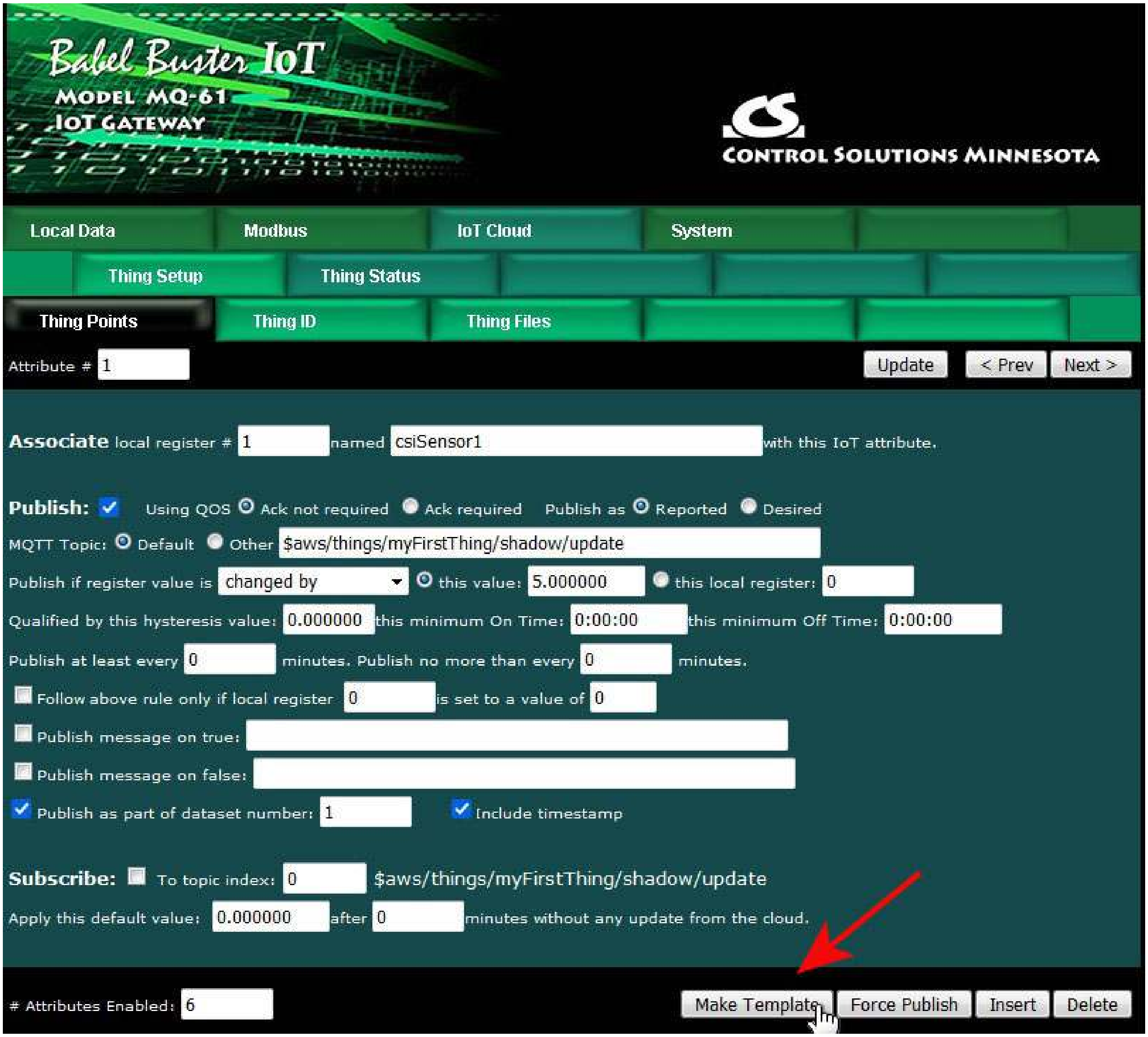
The Babel Buster IoT Gateway can generate that JSON document for you. Start by making sure you have configured all of the desired data points on your Thing Points list.



Showing attributes from 1

Atr #	Local Reg #	Attribute (Register) Name	Pub	Pub Ack	Sub	Periodic	Publish Condition	Reg	Threshold
1	1	csiSensor1	☑	☐	☐	0	changed by	☐	5.000000
2	2	csiSensor2	☑	☐	☐	0	changed by	☐	5.000000
3	3	csiSensor3	☑	☐	☐	0	changed by	☐	5.000000
4	4	csiSensor4	☑	☐	☐	0	changed by	☐	5.000000
5	5	csiSensor5	☑	☐	☐	0	changed by	☐	5.000000
6	0		☐	☐	☐	0	n/a	☐	0.000000

Click on the attribute number in the first column for any data point in the data set. Then click Make Template.



Attribute # 1

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 $aws/things/myFirstThing/shadow/update

Publish if register value is changed by ● this value: 5.000000 ○ 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:

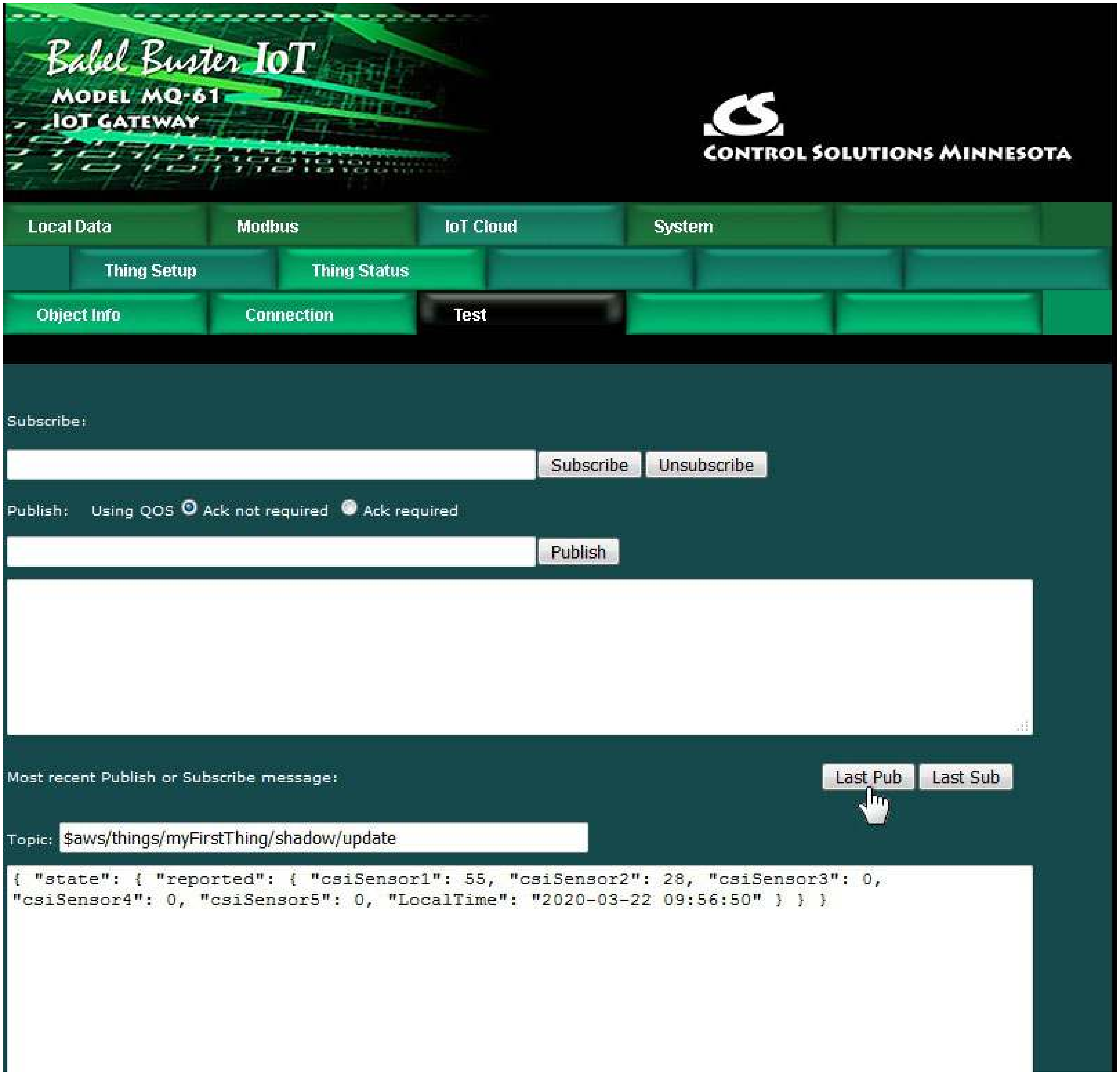
☑ Publish as part of dataset number: 1 ☑ Include timestamp

Subscribe: ☐ To topic index: 0 $aws/things/myFirstThing/shadow/update

Apply this default value: 0.000000 after 0 minutes without any update from the cloud.

Attributes Enabled: 6

Now go to the Thing Status :: Test page and your template will be found waiting for you in the Last Pub window. If you already have data being published, it is possible that your template got overwritten by another publish before you got here. Go back and re-click the Make Template button if the content does not appear to include the data points you expected.



Subscribe:

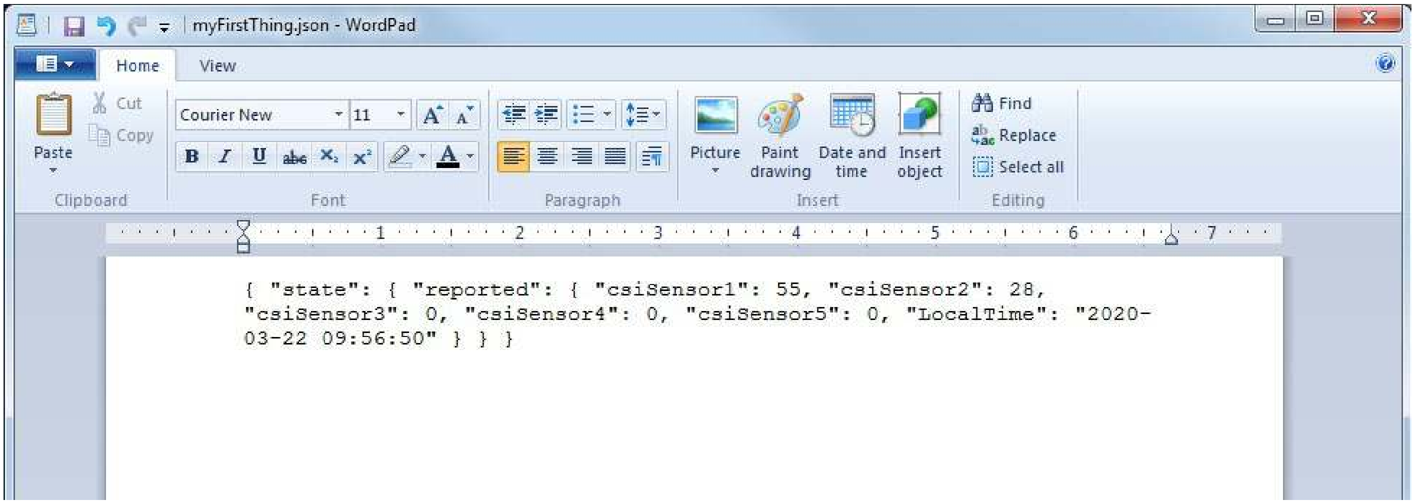
Publish: Using QOS ● Ack not required ○ Ack required

Most recent Publish or Subscribe message:

Topic: $aws/things/myFirstThing/shadow/update

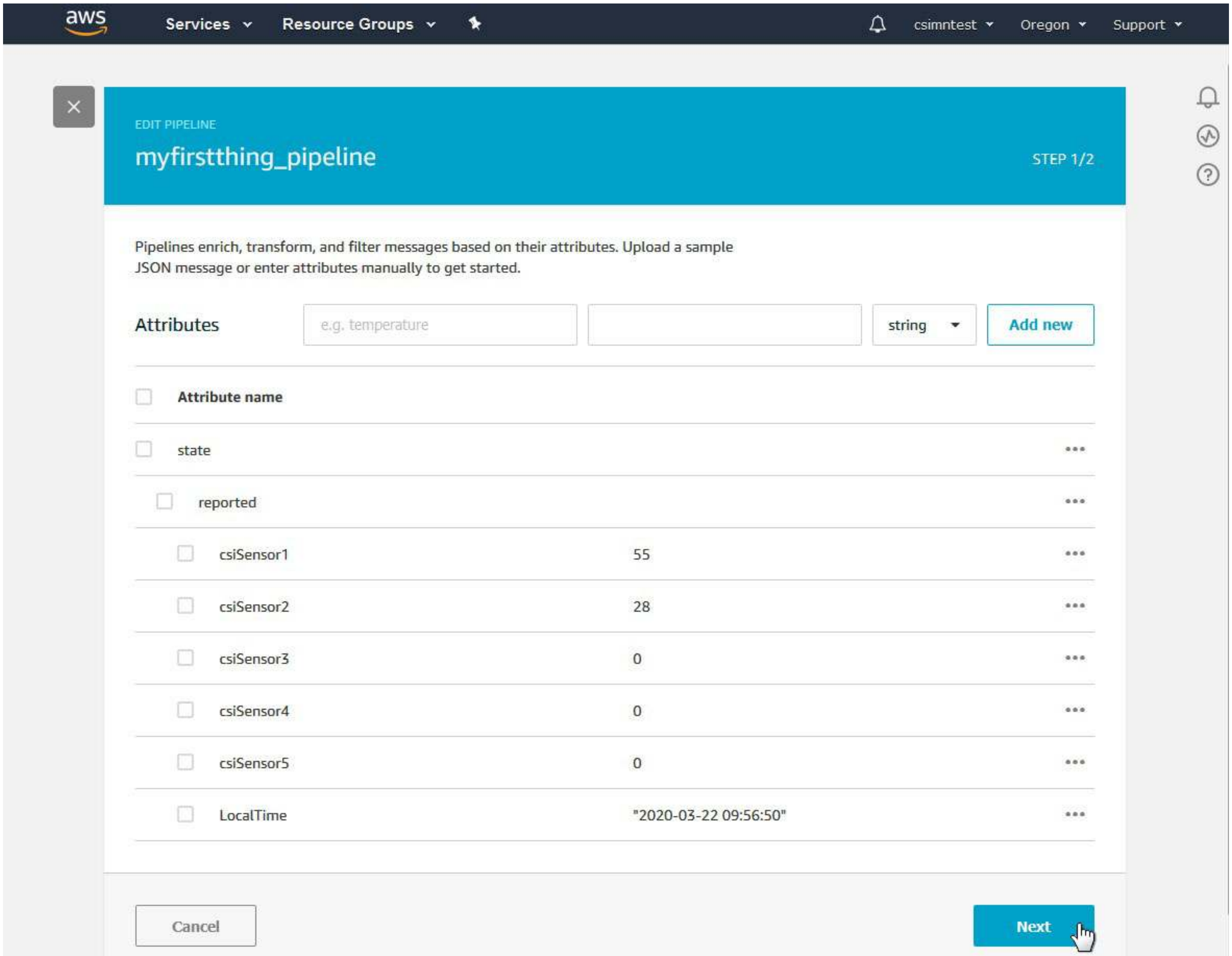
```
{ "state": { "reported": { "csiSensor1": 55, "csiSensor2": 28, "csiSensor3": 0,
"csiSensor4": 0, "csiSensor5": 0, "LocalTime": "2020-03-22 09:56:50" } } }
```

Copy the content to a blank text document, and then save that content as a file with a .json suffix.



```
{ "state": { "reported": { "csiSensor1": 55, "csiSensor2": 28,
"csiSensor3": 0, "csiSensor4": 0, "csiSensor5": 0, "LocalTime": "2020-
03-22 09:56:50" } } }
```

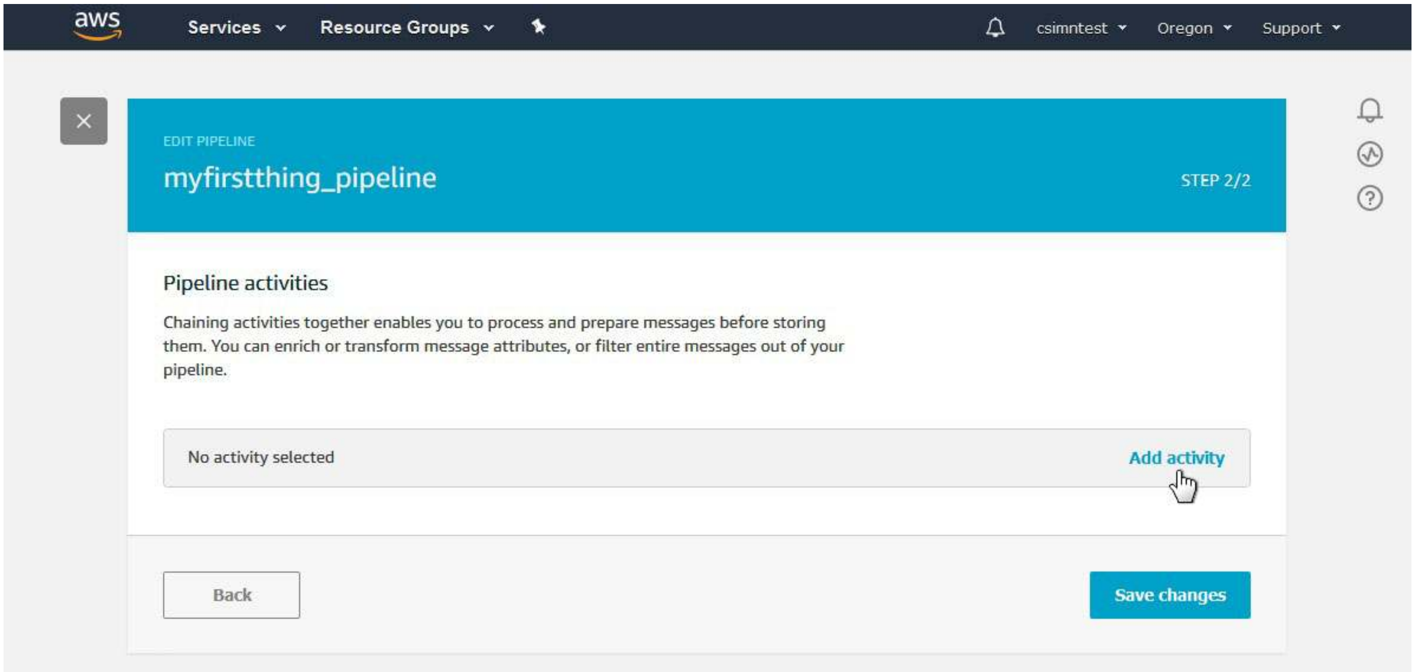
Now click the "uploading a JSON document" link pictured above and upload the *.json file you just created. The result will be to display that template in the form illustrated below. Click Next to continue.



Pipelines enrich, transform, and filter messages based on their attributes. Upload a sample JSON message or enter attributes manually to get started.

□ Attribute name	
□ state	
□ reported	
□ csiSensor1	55
□ csiSensor2	28
□ csiSensor3	0
□ csiSensor4	0
□ csiSensor5	0
□ LocalTime	"2020-03-22 09:56:50"

You have told the pipeline what to look for. The next step is to add an activity (actually we will be adding two). Click Add activity.

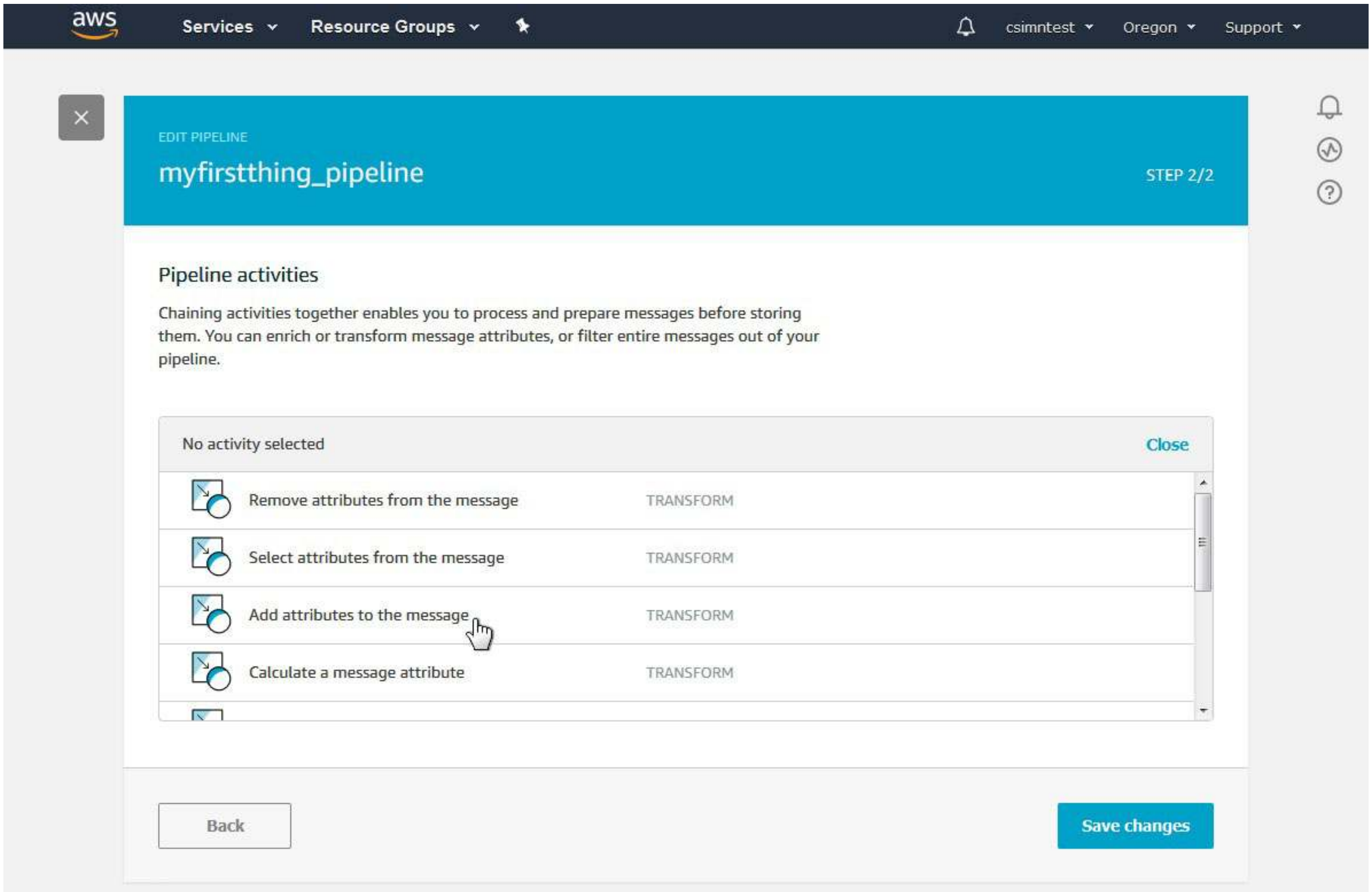


Pipeline activities

Chaining activities together enables you to process and prepare messages before storing them. You can enrich or transform message attributes, or filter entire messages out of your pipeline.

No activity selected

The first activity you want to select is "Add attributes to the message".



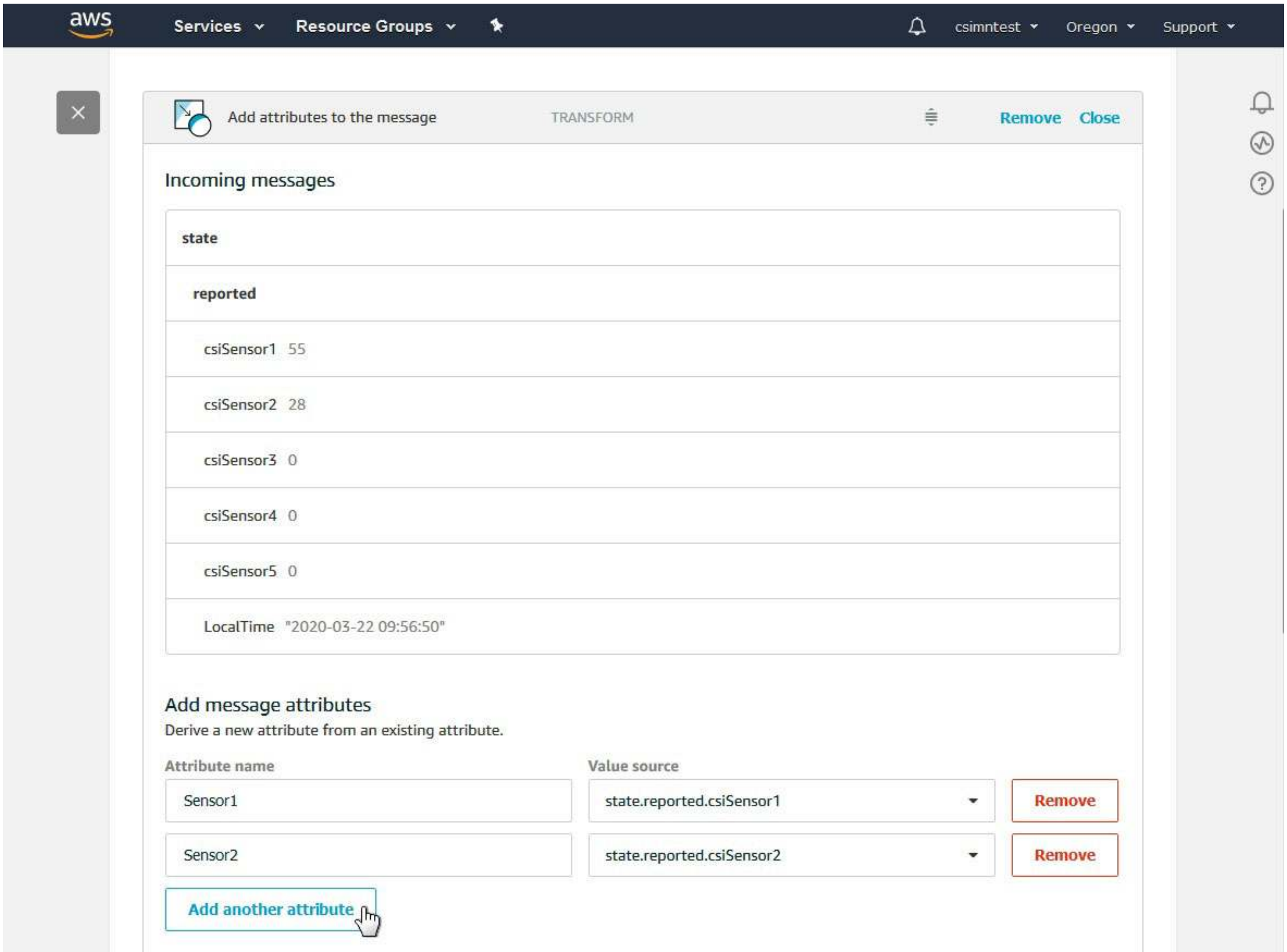
Pipeline activities

Chaining activities together enables you to process and prepare messages before storing them. You can enrich or transform message attributes, or filter entire messages out of your pipeline.

No activity selected

Remove attributes from the message	TRANSFORM
Select attributes from the message	TRANSFORM
Add attributes to the message	TRANSFORM
Calculate a message attribute	TRANSFORM

We want to add message attributes - one per data point. The purpose of this is to give us the opportunity to pass the data through without all of the JSON stuff coming with the data. So add attributes with some name that you will want to see in future visualizations of the data, and select the source from the state.



Add attributes to the message

Incoming messages

state

reported

csiSensor1 55

csiSensor2 28

csiSensor3 0

csiSensor4 0

csiSensor5 0

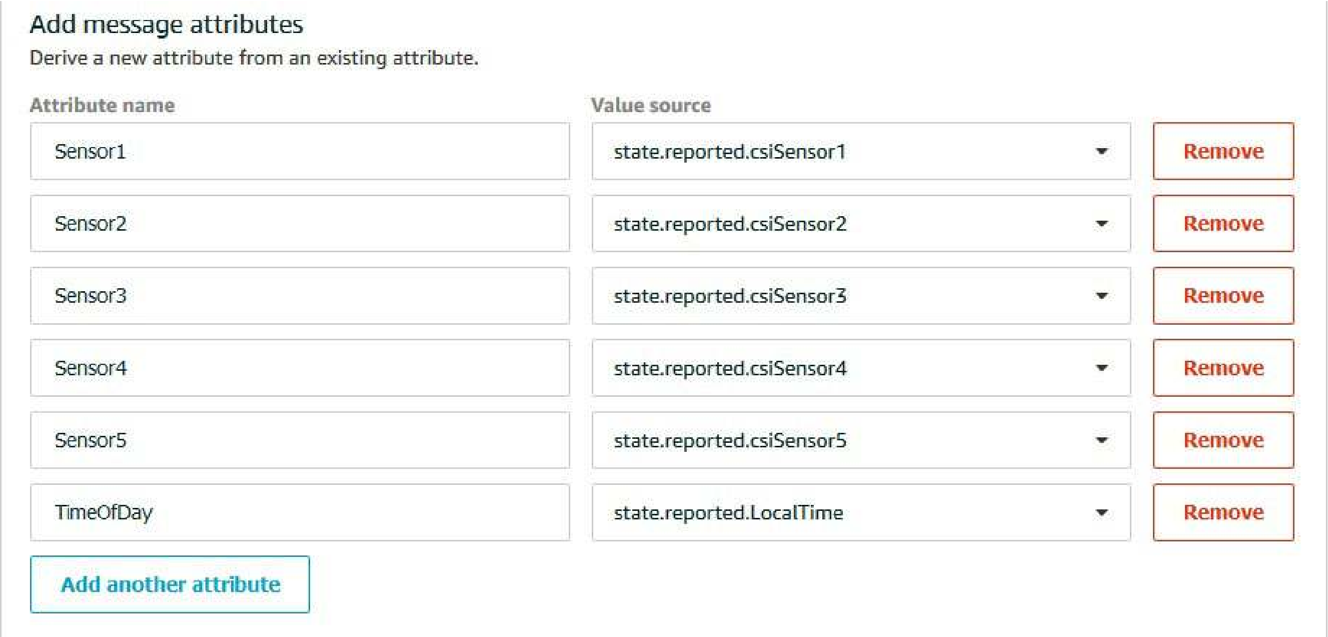
LocalTime "2020-03-22 09:56:50"

Add message attributes

Derive a new attribute from an existing attribute.

Attribute name	Value source
Sensor1	state.reported.csiSensor1
Sensor2	state.reported.csiSensor2

The complete set of added attributes in this example is illustrated below.

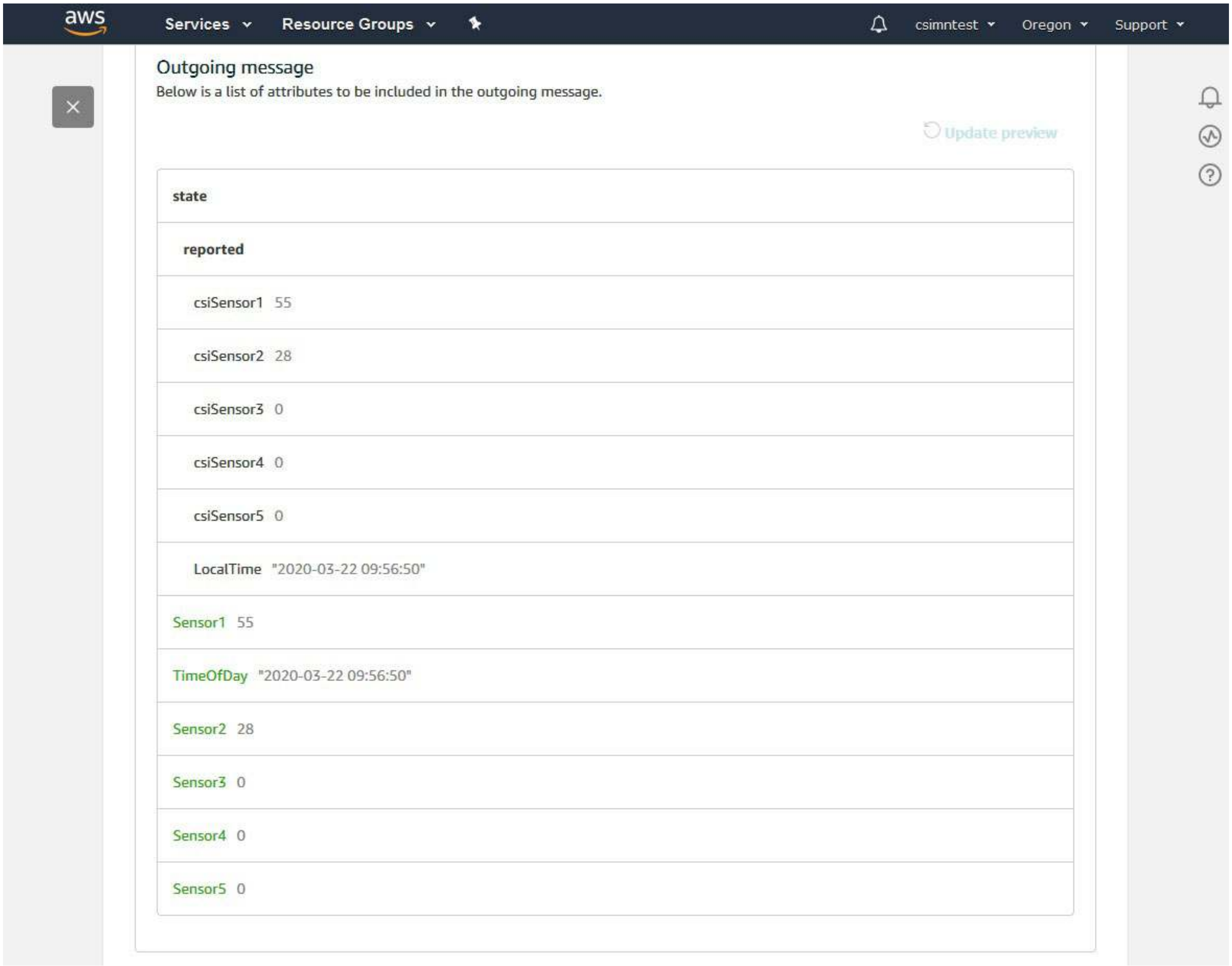


Add message attributes

Derive a new attribute from an existing attribute.

Attribute name	Value source
Sensor1	state.reported.csiSensor1
Sensor2	state.reported.csiSensor2
Sensor3	state.reported.csiSensor3
Sensor4	state.reported.csiSensor4
Sensor5	state.reported.csiSensor5
TimeOfDay	state.reported.LocalTime

At this point, the outgoing message will appear as shown below. Click on "Update preview" to get an updated preview after making changes. Once updated, and until more changes are made, the "Update preview" is unavailable.

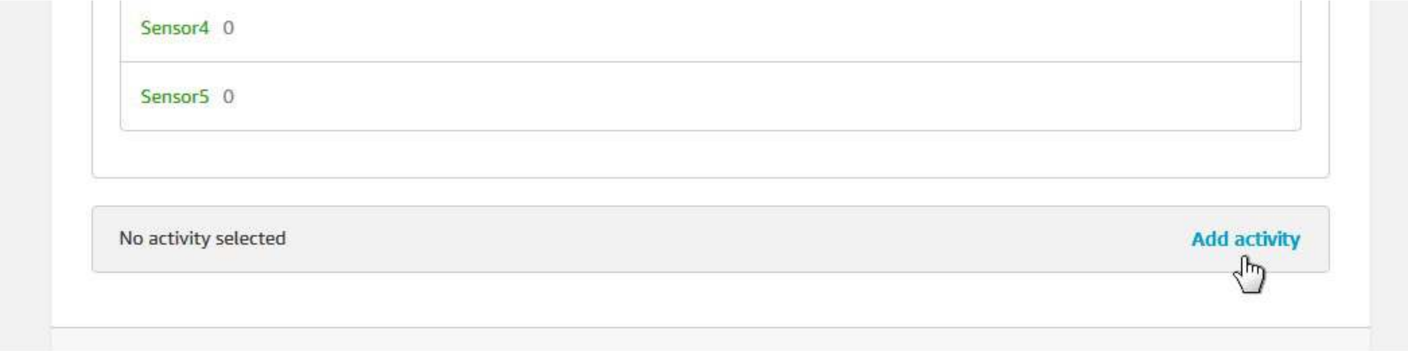


Outgoing message

Below is a list of attributes to be included in the outgoing message.

state	
reported	
csiSensor1	55
csiSensor2	28
csiSensor3	0
csiSensor4	0
csiSensor5	0
LocalTime	"2020-03-22 09:56:50"
Sensor1	55
TimeOfDay	"2020-03-22 09:56:50"
Sensor2	28
Sensor3	0
Sensor4	0
Sensor5	0

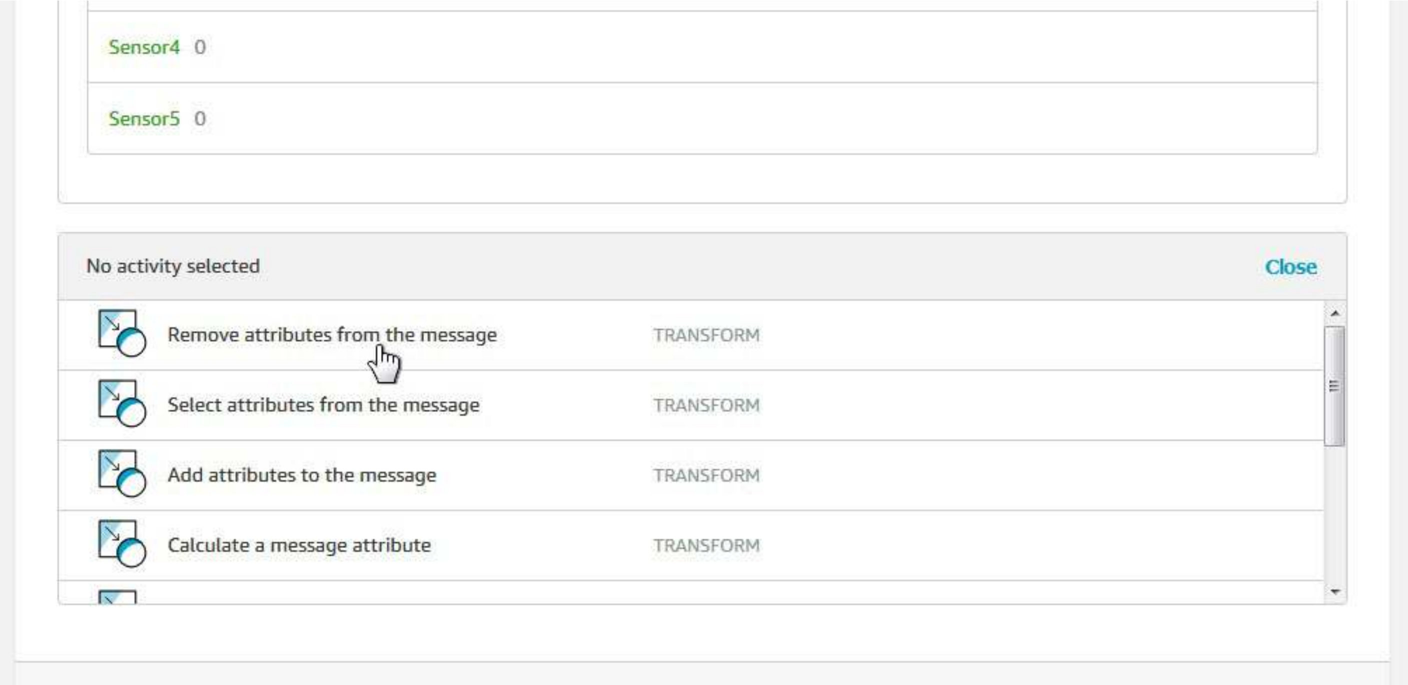
Now click "Add activity" to add another activity.



Sensor4	0
Sensor5	0

No activity selected

This time, select "Remove attributes from the message".



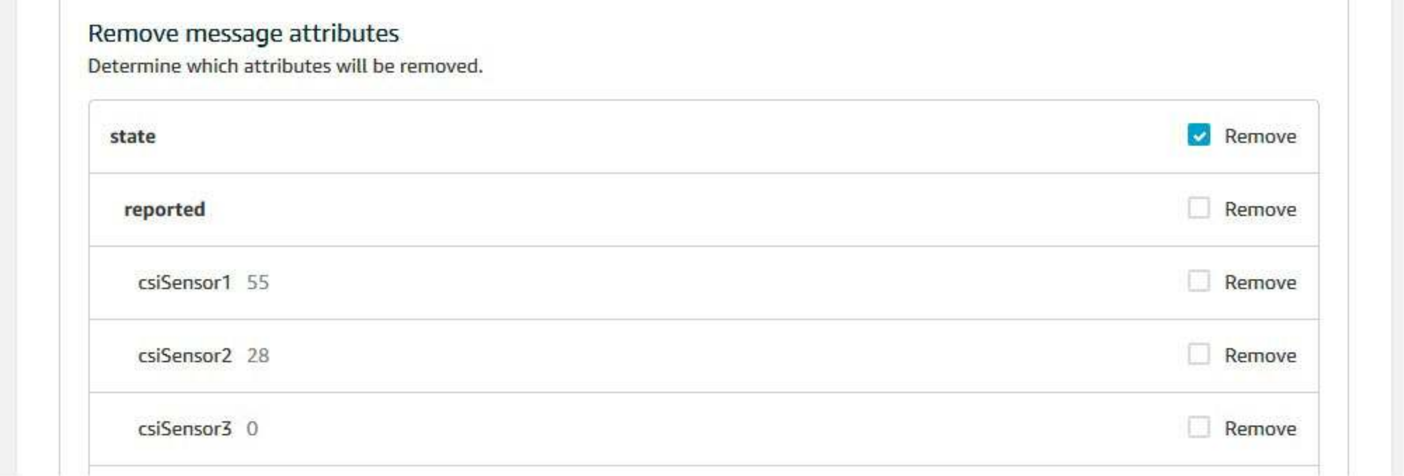
Sensor4 0

Sensor5 0

No activity selected Close

Remove attributes from the message	TRANSFORM
Select attributes from the message	TRANSFORM
Add attributes to the message	TRANSFORM
Calculate a message attribute	TRANSFORM

Check the box on the "state" line. We are going to remove the entire JSON object.

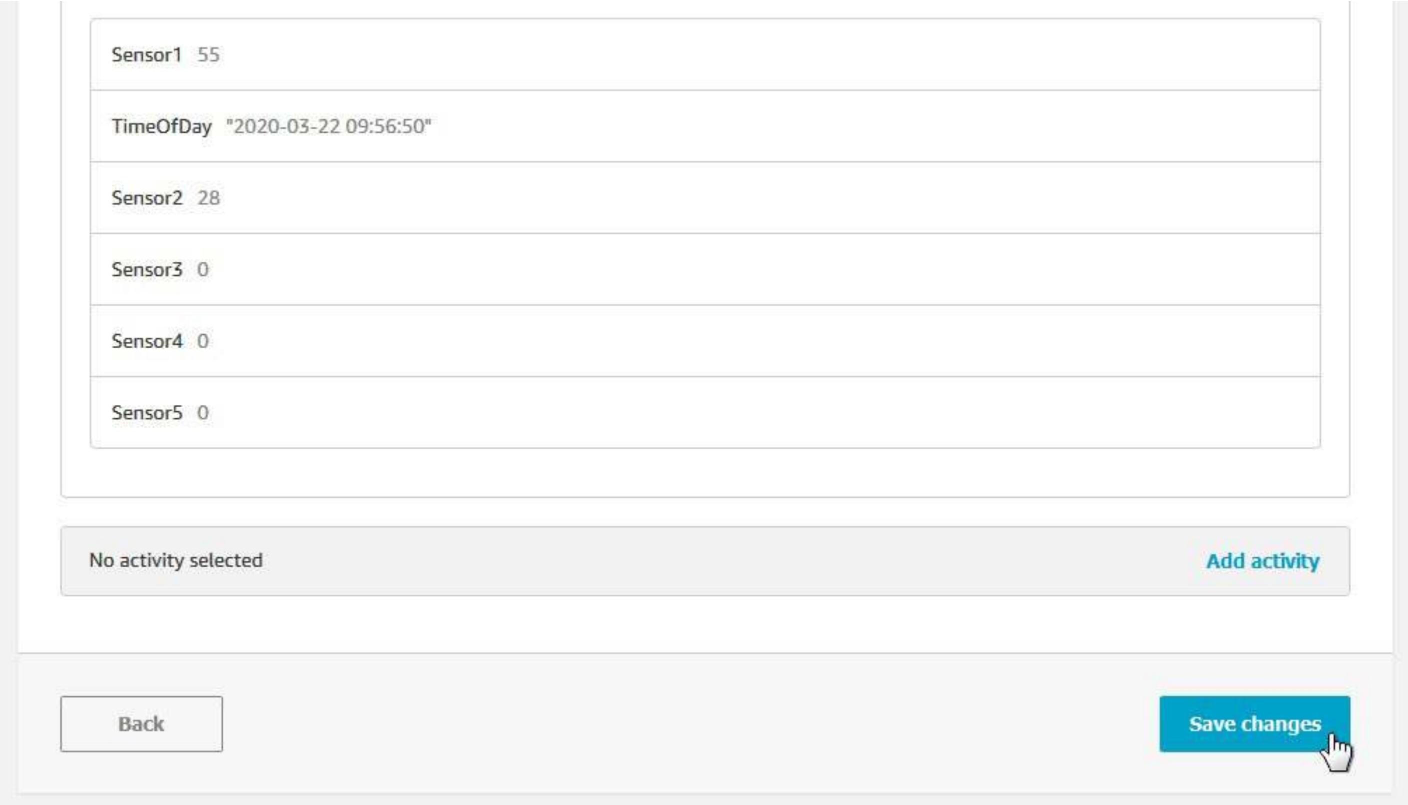


Remove message attributes

Determine which attributes will be removed.

state	☑ Remove
reported	□ Remove
csiSensor1 55	□ Remove
csiSensor2 28	□ Remove
csiSensor3 0	□ Remove

The outgoing message now looks like the following. Click Save changes.



Sensor1 55

TimeOfDay "2020-03-22 09:56:50"

Sensor2 28

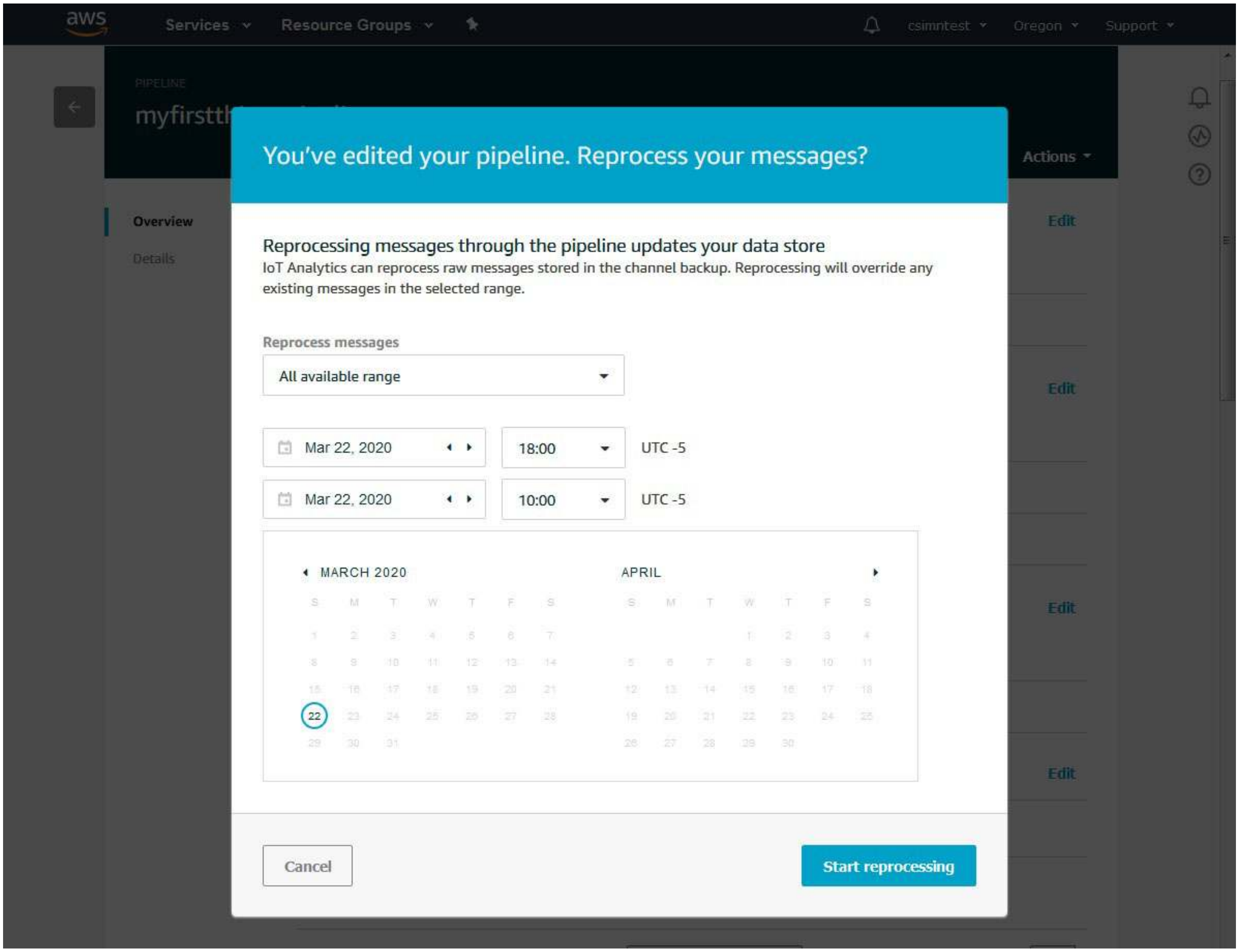
Sensor3 0

Sensor4 0

Sensor5 0

No activity selected

AWS will automatically display the following window at this point. If you do not yet have any data published to this pipeline, your answer here doesn't really matter. If you have come back later and made changes and want to see the effect of those changes, then you will want to reprocess messages.



You've edited your pipeline. Reprocess your messages?

Reprocessing messages through the pipeline updates your data store

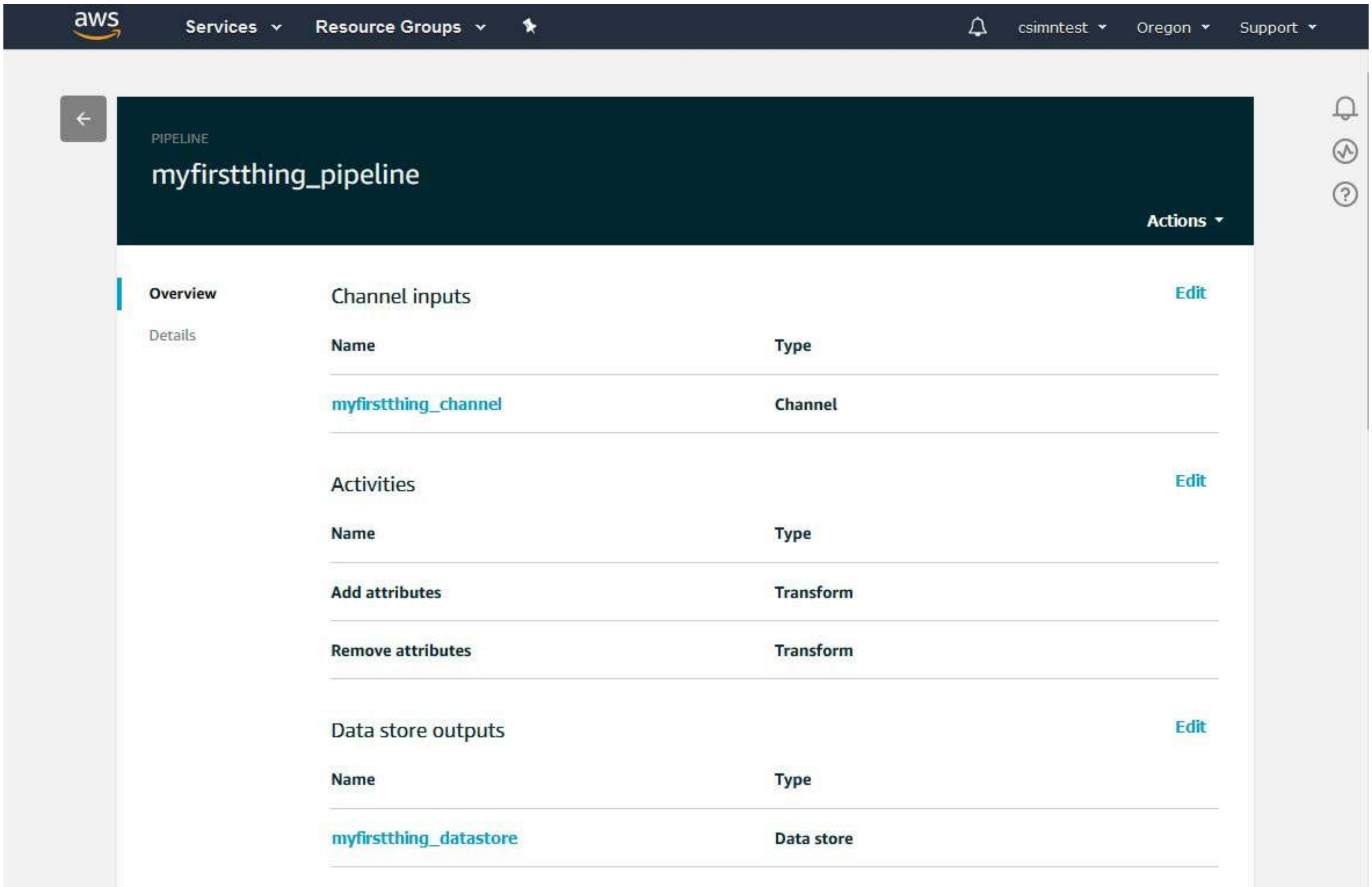
IoT Analytics can reprocess raw messages stored in the channel backup. Reprocessing will override any existing messages in the selected range.

Reprocess messages: All available range

Mar 22, 2020 18:00 UTC -5

Mar 22, 2020 10:00 UTC -5

The pipeline setup is now complete, and will appear as follows.



PIPELINE

myfirstthing_pipeline

Channel inputs

Name	Type
myfirstthing_channel	Channel

Activities

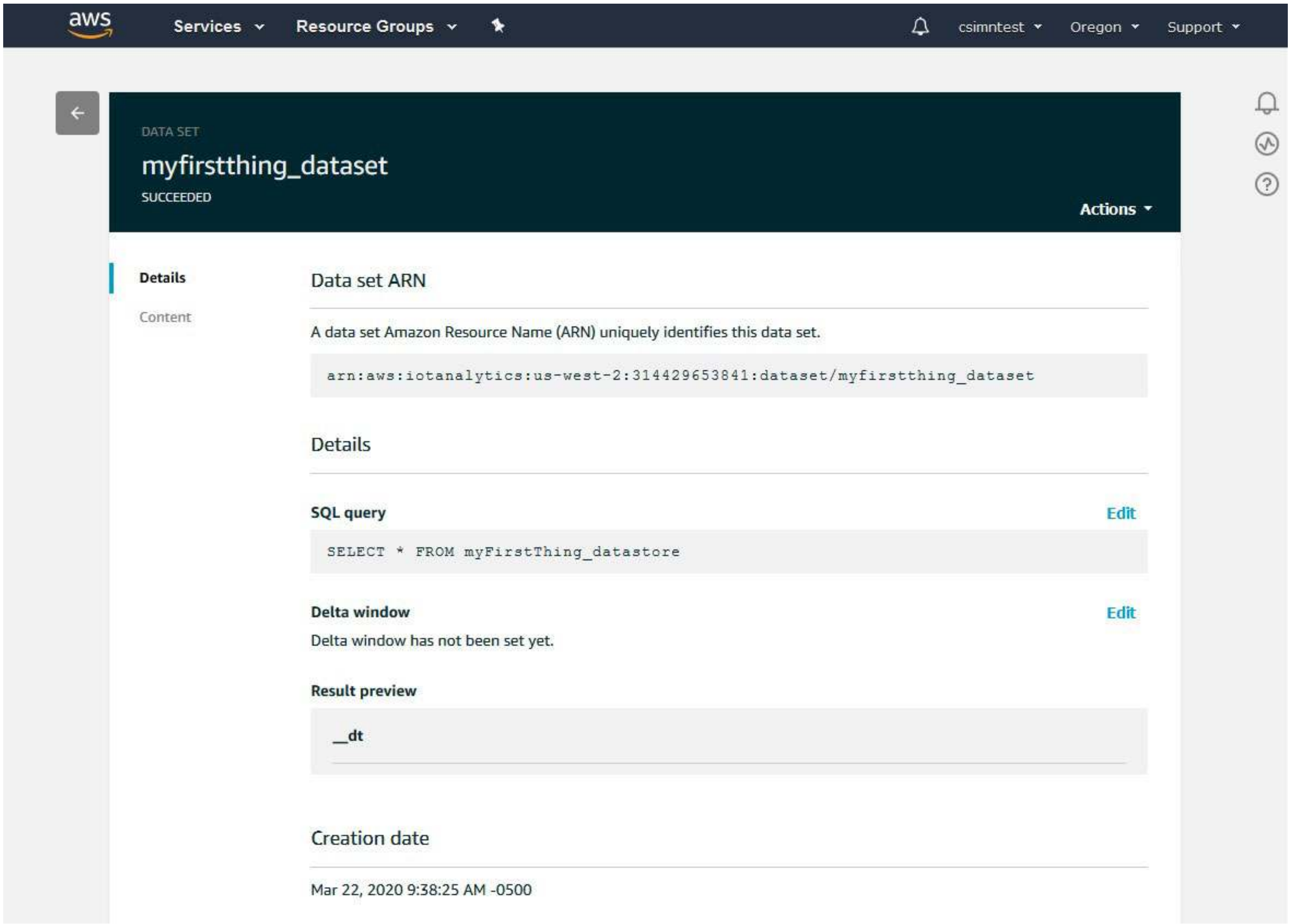
Name	Type
Add attributes	Transform
Remove attributes	Transform

Data store outputs

Name	Type
myfirstthing_datastore	Data store

17.3 Review Data set

Click on Data sets on the main IoT Analytics menu. There is not much that needs to be set up with the data set.



myfirstthing_dataset

SUCCEEDED

Data set ARN

A data set Amazon Resource Name (ARN) uniquely identifies this data set.

```
arn:aws:iotanalytics:us-west-2:314429653841:dataset/myfirstthing_dataset
```

Details

SQL query

```
SELECT * FROM myFirstThing_datastore
```

Delta window

Delta window has not been set yet.

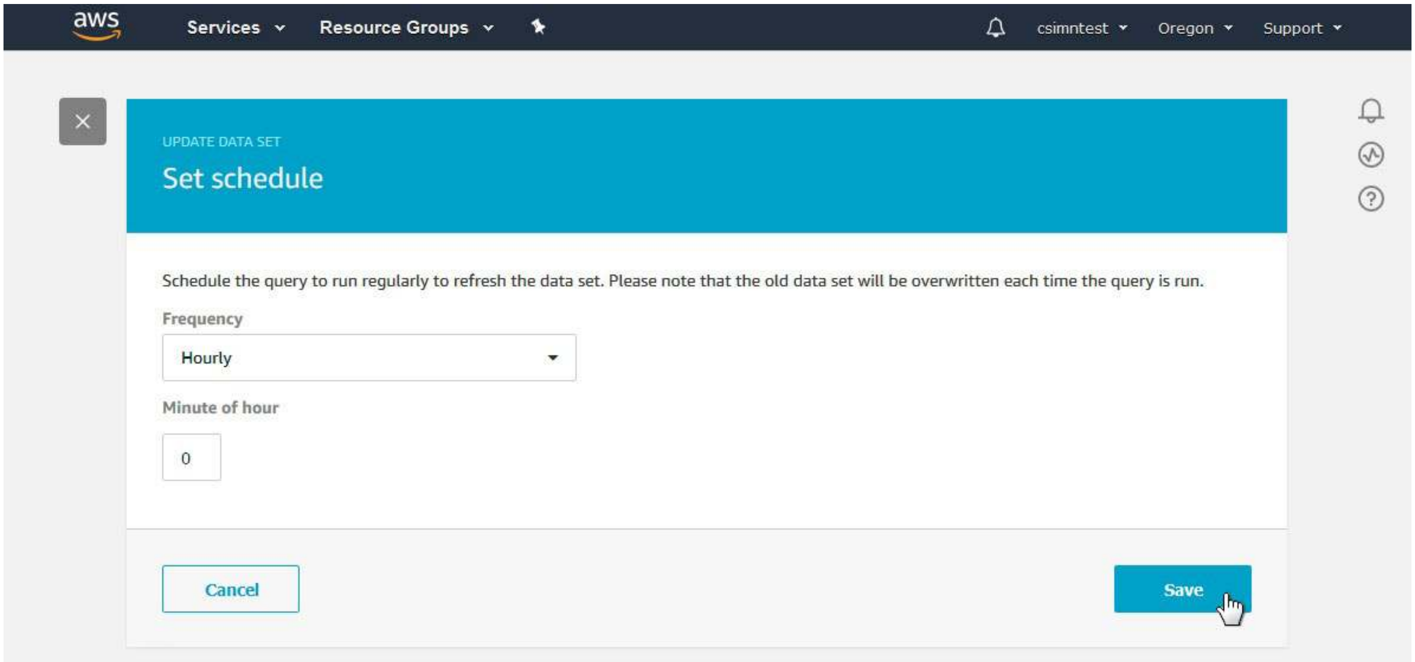
Result preview

__dt

Creation date

Mar 22, 2020 9:38:25 AM -0500

The data set will only update when you manually tell it to do so unless you set an update schedule. Click on Edit next to Schedule, and select an update period. Then click Save.



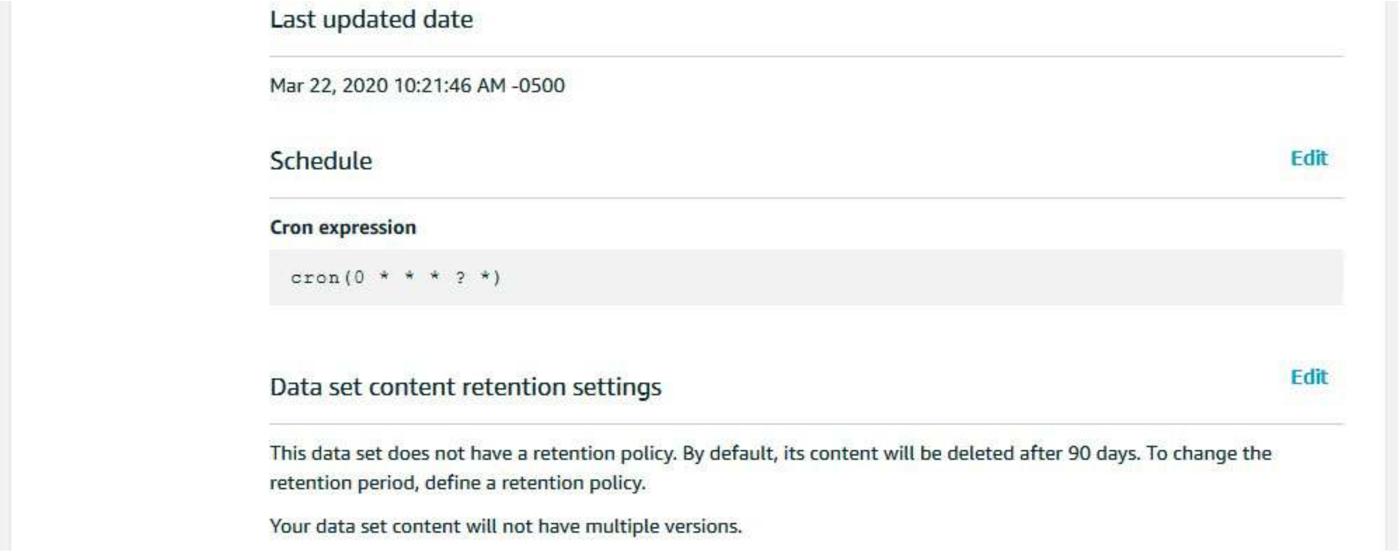
Set schedule

Schedule the query to run regularly to refresh the data set. Please note that the old data set will be overwritten each time the query is run.

Frequency: Hourly

Minute of hour: 0

The schedule is displayed in Linux Cron format when you are done.



Last updated date

Mar 22, 2020 10:21:46 AM -0500

Schedule

Cron expression

```
cron(0 * * * ? *)
```

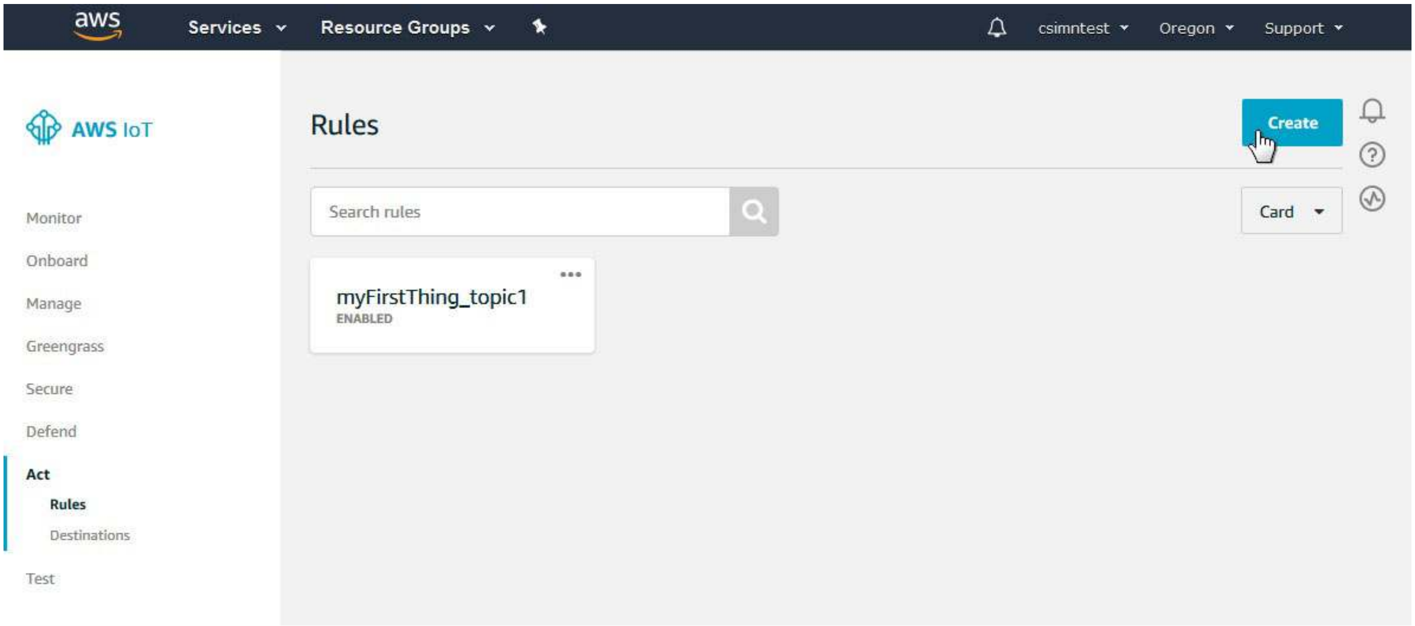
Data set content retention settings

This data set does not have a retention policy. By default, its content will be deleted after 90 days. To change the retention period, define a retention policy.

Your data set content will not have multiple versions.

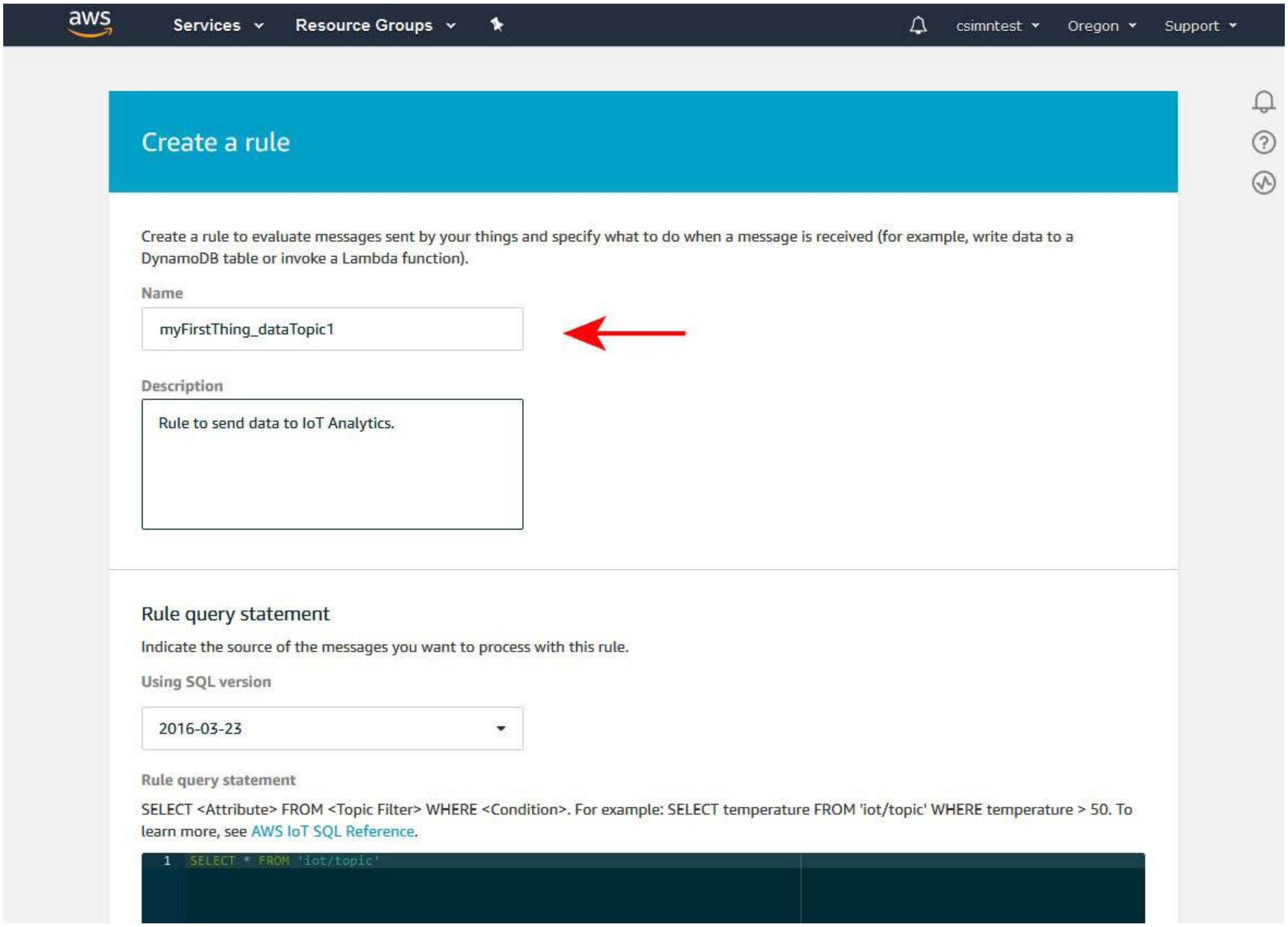
17.4 Create Action Rule for IoT Analytics

Everything is now set up for IoT Analytics to receive your data. Next, you need to go back to the IoT Core and tell your Thing to send data to IoT Analytics. This is done be creating another Action rule. Go to Rules, and click Create.



myFirstThing_topic1
ENABLED

Provide a name for this Action rule and provide a description for your own reference.



Create a rule

Create a rule to evaluate messages sent by your things and specify what to do when a message is received (for example, write data to a DynamoDB table or invoke a Lambda function).

Name

myFirstThing_dataTopic1

Description

Rule to send data to IoT Analytics.

Rule query statement

Indicate the source of the messages you want to process with this rule.

Using SQL version

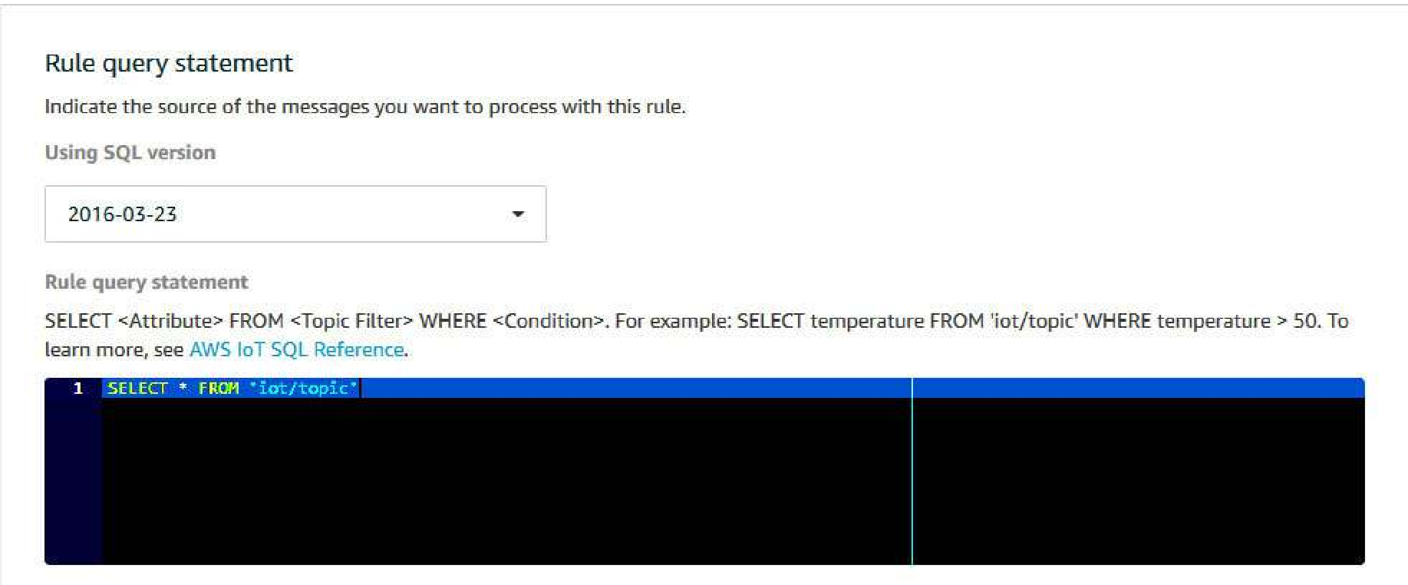
2016-03-23

Rule query statement

SELECT <Attribute> FROM <Topic Filter> WHERE <Condition>. For example: SELECT temperature FROM 'iot/topic' WHERE temperature > 50. To learn more, see AWS IoT SQL Reference.

```
SELECT * FROM 'iot/topic'
```

Data is selected using an SQL style query. The 'iot/topic' that it starts out with is just a place holder.



Rule query statement

Indicate the source of the messages you want to process with this rule.

Using SQL version

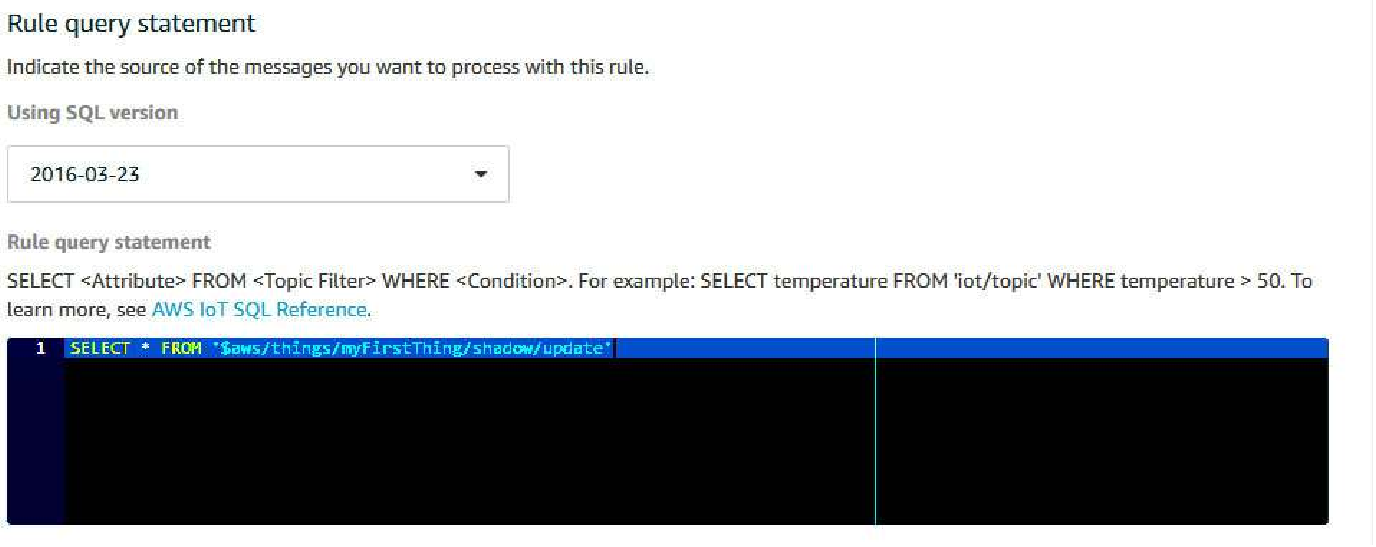
2016-03-23

Rule query statement

SELECT <Attribute> FROM <Topic Filter> WHERE <Condition>. For example: SELECT temperature FROM 'iot/topic' WHERE temperature > 50. To learn more, see AWS IoT SQL Reference.

```
SELECT * FROM 'iot/topic'
```

The SQL query needs to contain the topic that your Thing will be publishing to. The default topic is '$aws/things/myFirstThing/shadow/update' (where 'myFirstThing' will be whatever you named your thing).



Rule query statement

Indicate the source of the messages you want to process with this rule.

Using SQL version

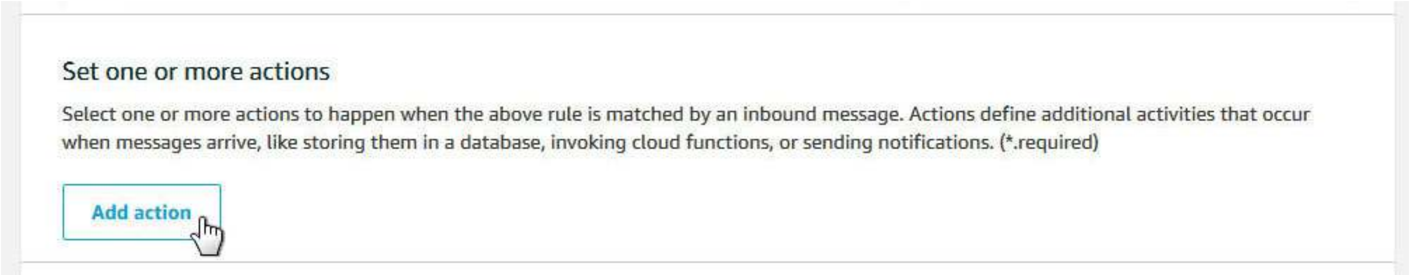
2016-03-23

Rule query statement

SELECT <Attribute> FROM <Topic Filter> WHERE <Condition>. For example: SELECT temperature FROM 'iot/topic' WHERE temperature > 50. To learn more, see AWS IoT SQL Reference.

```
SELECT * FROM '$aws/things/myFirstThing/shadow/update'
```

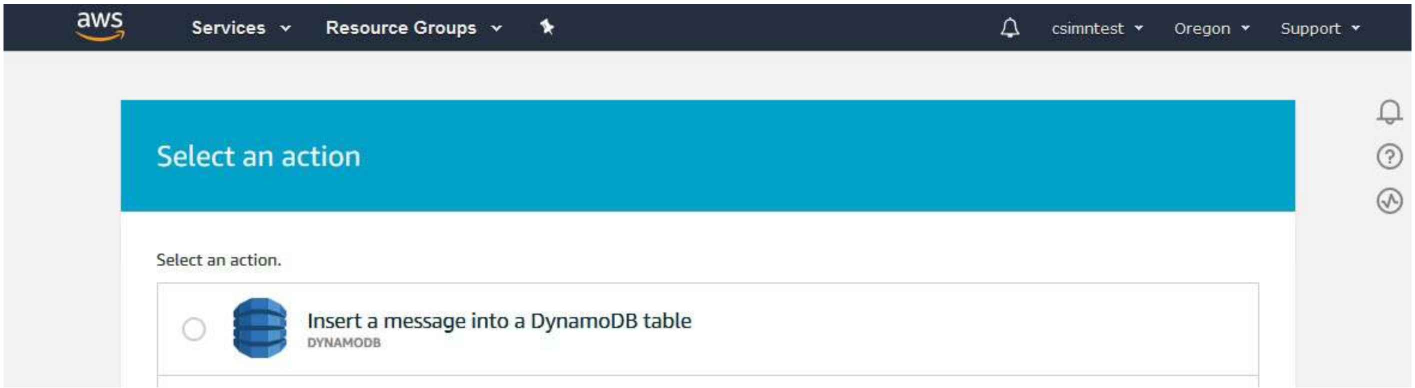
After editing the SQL query to reflect your topic, click Add action.



Set one or more actions

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. (*.required)

The Action list will appear.

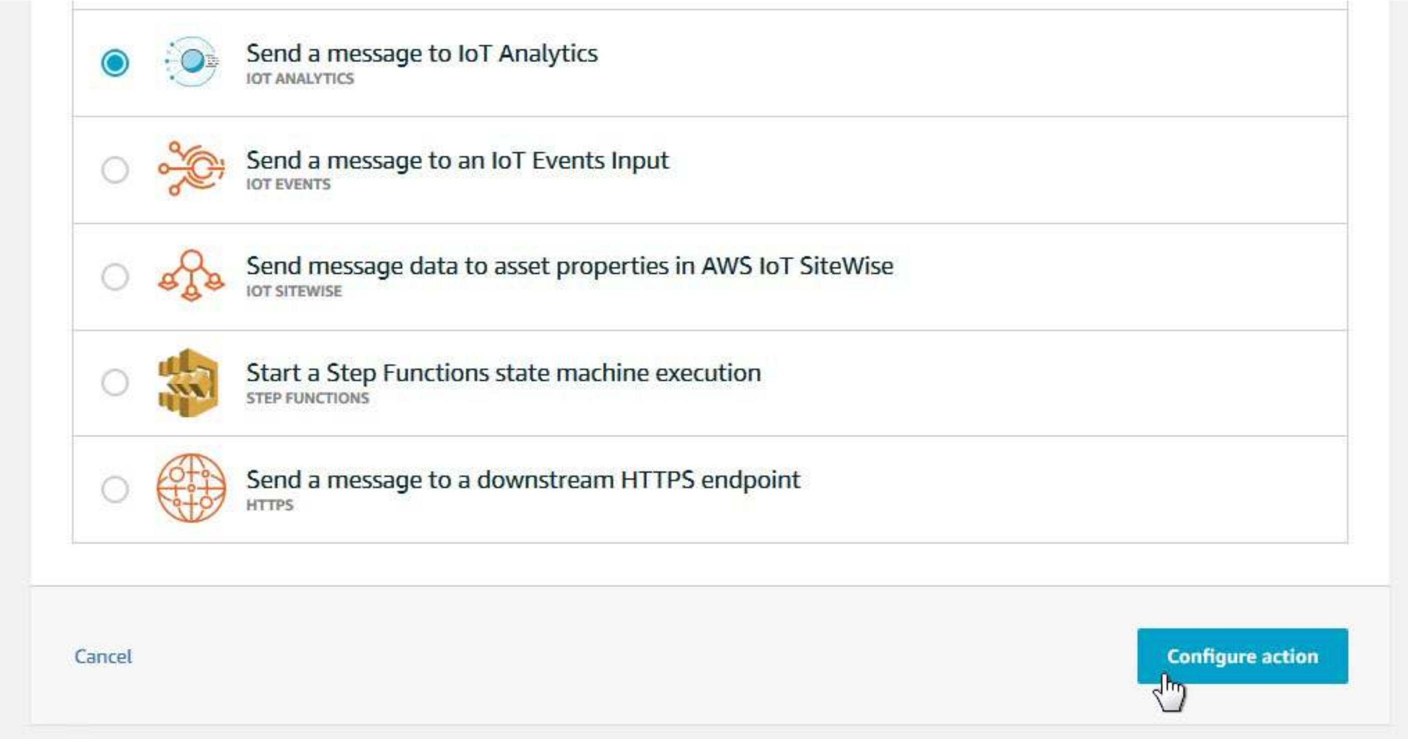


Select an action

Select an action.

○ Insert a message into a DynamoDB table
DYNAMODB

Scroll down to "Send a message to IoT Analytics", and select that option. Then click Configure action.



● Send a message to IoT Analytics
IOT ANALYTICS

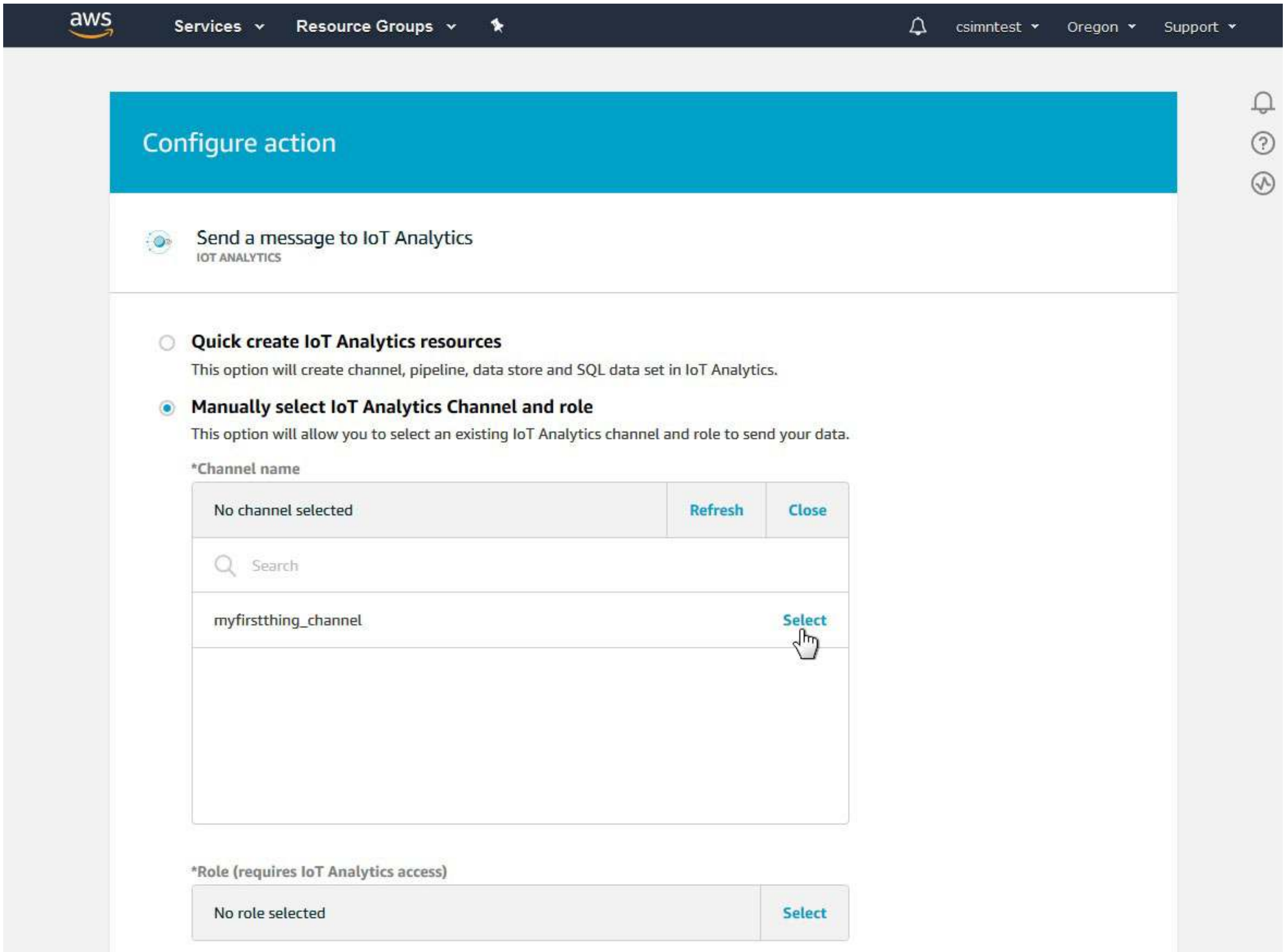
○ Send a message to an IoT Events Input
IOT EVENTS

○ Send message data to asset properties in AWS IoT SiteWise
IOT SITEWISE

○ Start a Step Functions state machine execution
STEP FUNCTIONS

○ Send a message to a downstream HTTPS endpoint
HTTPS

The IoT Analytics channel previously created should appear on this list. Select it.



Configure action

Send a message to IoT Analytics
IOT ANALYTICS

○ **Quick create IoT Analytics resources**
This option will create channel, pipeline, data store and SQL data set in IoT Analytics.

● **Manually select IoT Analytics Channel and role**
This option will allow you to select an existing IoT Analytics channel and role to send your data.

*Channel name

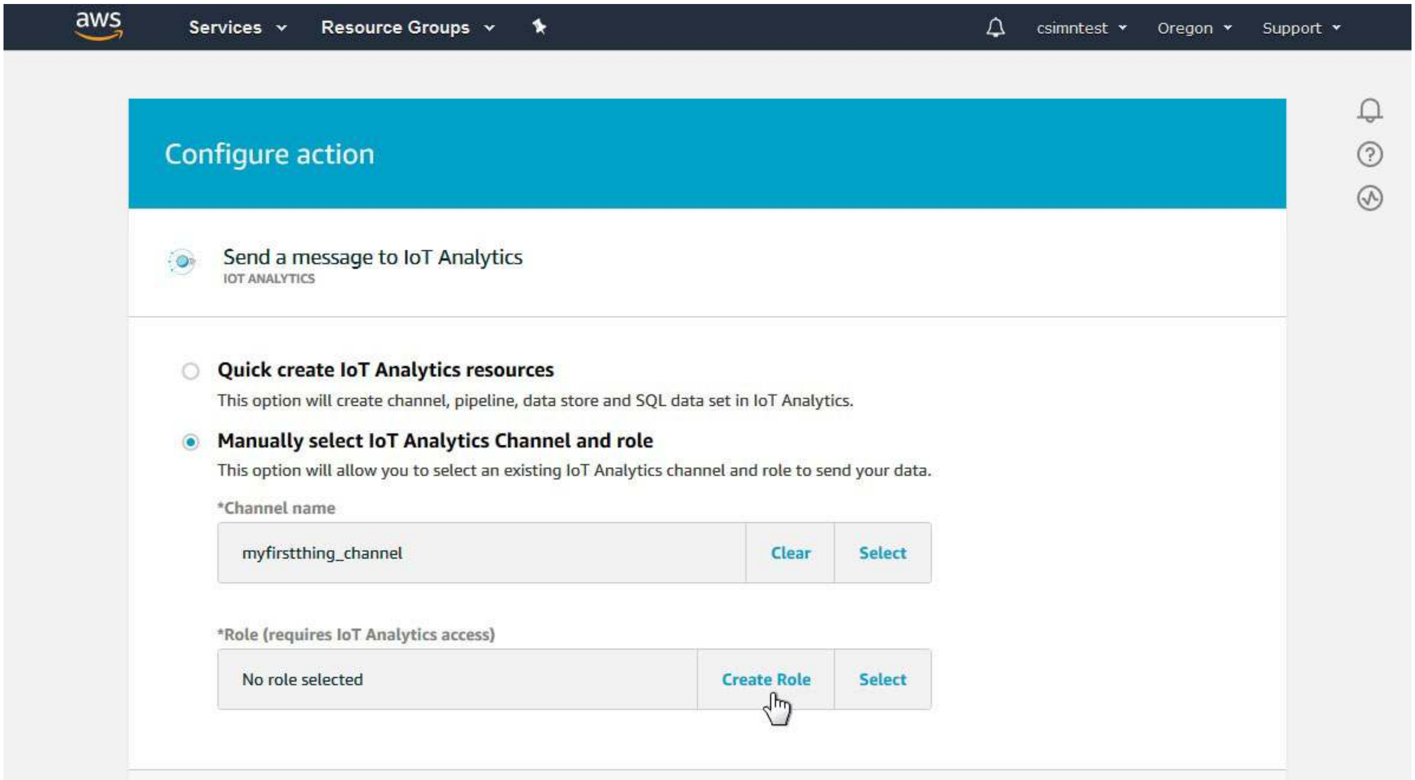
No channel selected

myfirstthing_channel

*Role (requires IoT Analytics access)

No role selected

You will need to create a role. Click Create Role.



Configure action

Send a message to IoT Analytics
IOT ANALYTICS

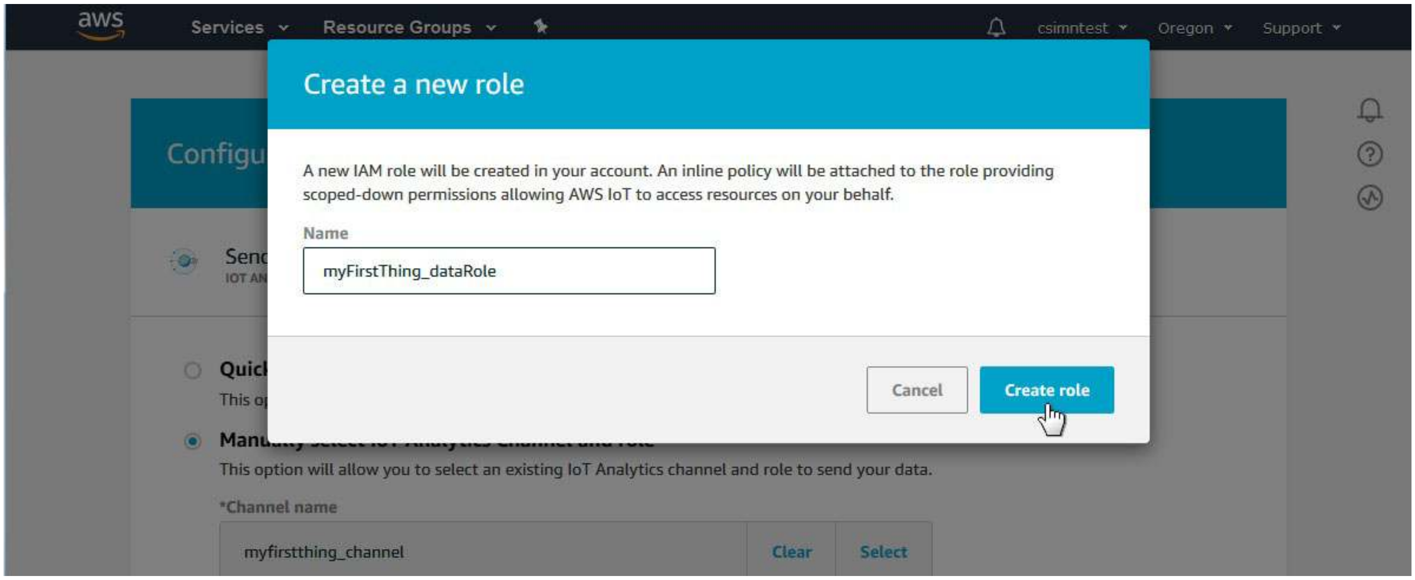
○ Quick create IoT Analytics resources
This option will create channel, pipeline, data store and SQL data set in IoT Analytics.

● Manually select IoT Analytics Channel and role
This option will allow you to select an existing IoT Analytics channel and role to send your data.

*Channel name: myfirstthing_channel

*Role (requires IoT Analytics access): No role selected

Provide a name for your role, and click Create role.

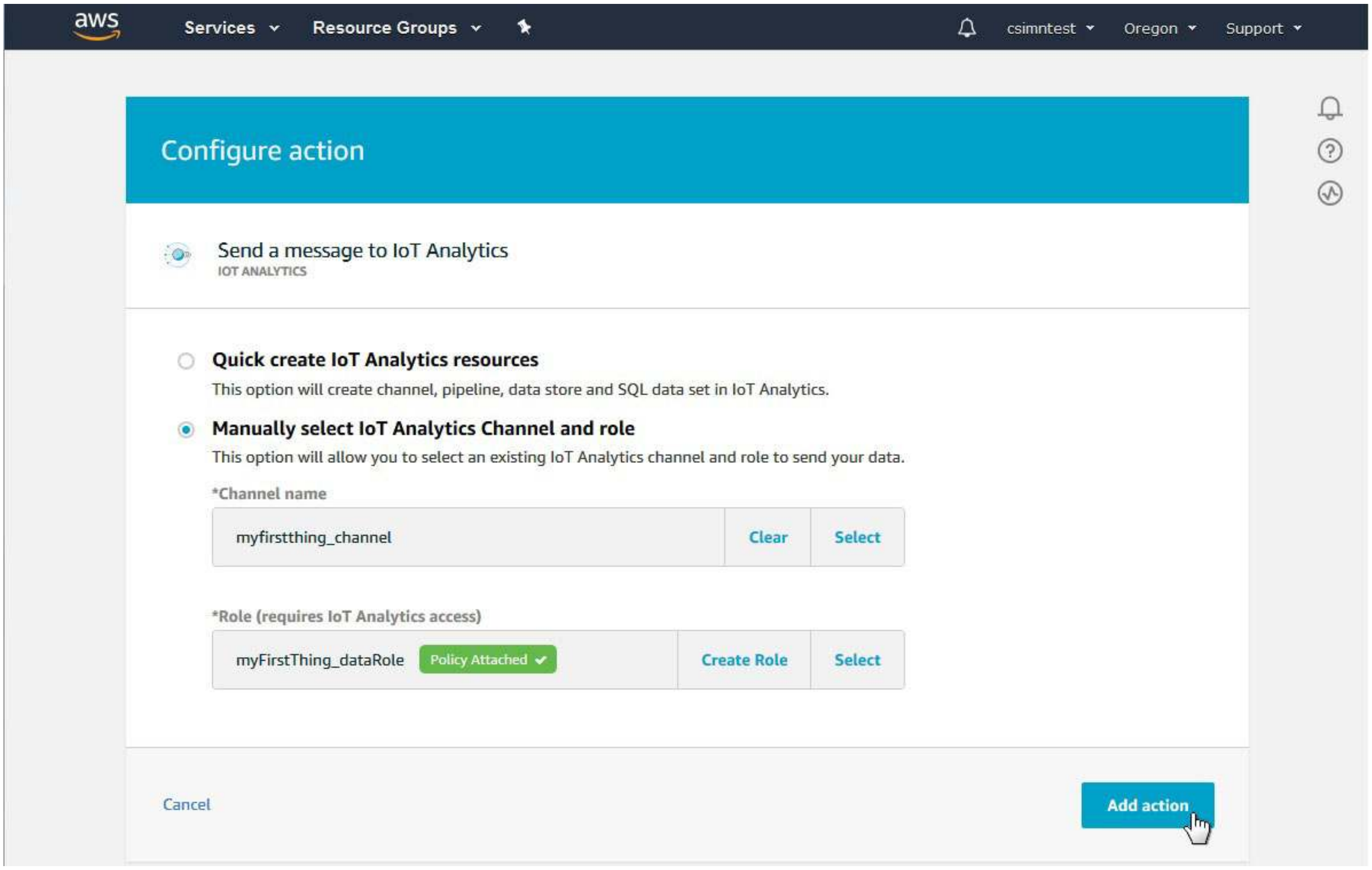


Create a new role

A new IAM role will be created in your account. An inline policy will be attached to the role providing scoped-down permissions allowing AWS IoT to access resources on your behalf.

Name: myFirstThing_dataRole

The action rule should now appear as follows. Click Add action.



Configure action

Send a message to IoT Analytics
IOT ANALYTICS

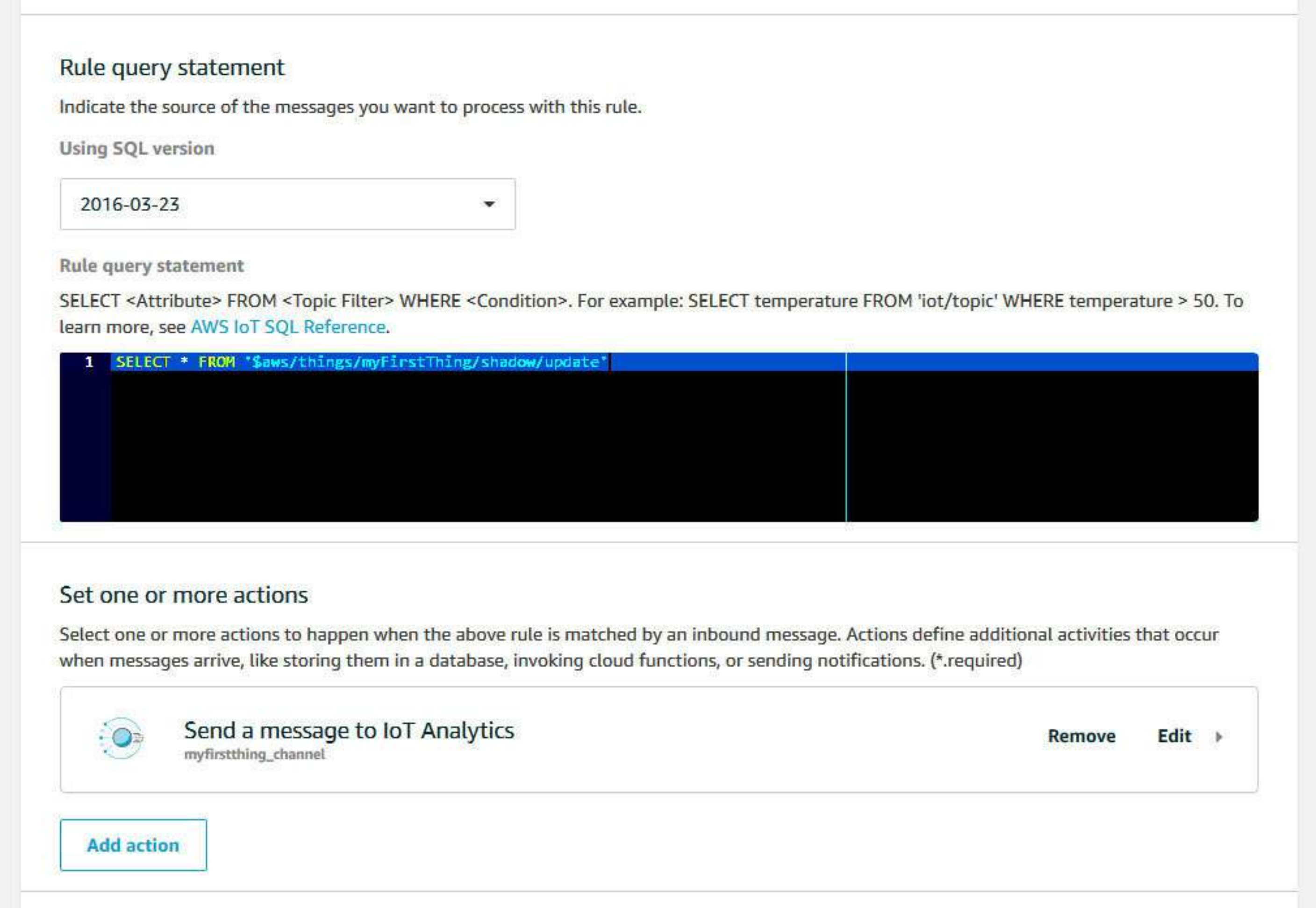
○ **Quick create IoT Analytics resources**
This option will create channel, pipeline, data store and SQL data set in IoT Analytics.

● **Manually select IoT Analytics Channel and role**
This option will allow you to select an existing IoT Analytics channel and role to send your data.

*Channel name: myfirstthing_channel

*Role (requires IoT Analytics access): myFirstThing_dataRole Policy Attached ✓

The finished Action rule will now look like this. Click Create rule.



Rule query statement

Indicate the source of the messages you want to process with this rule.

Using SQL version

2016-03-23

Rule query statement

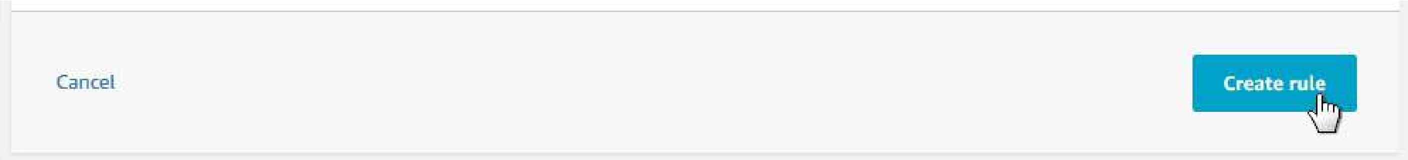
SELECT <Attribute> FROM <Topic Filter> WHERE <Condition>. For example: SELECT temperature FROM 'iot/topic' WHERE temperature > 50. To learn more, see AWS IoT SQL Reference.

```
SELECT * FROM '$aws/things/myFirstThing/shadow/update'
```

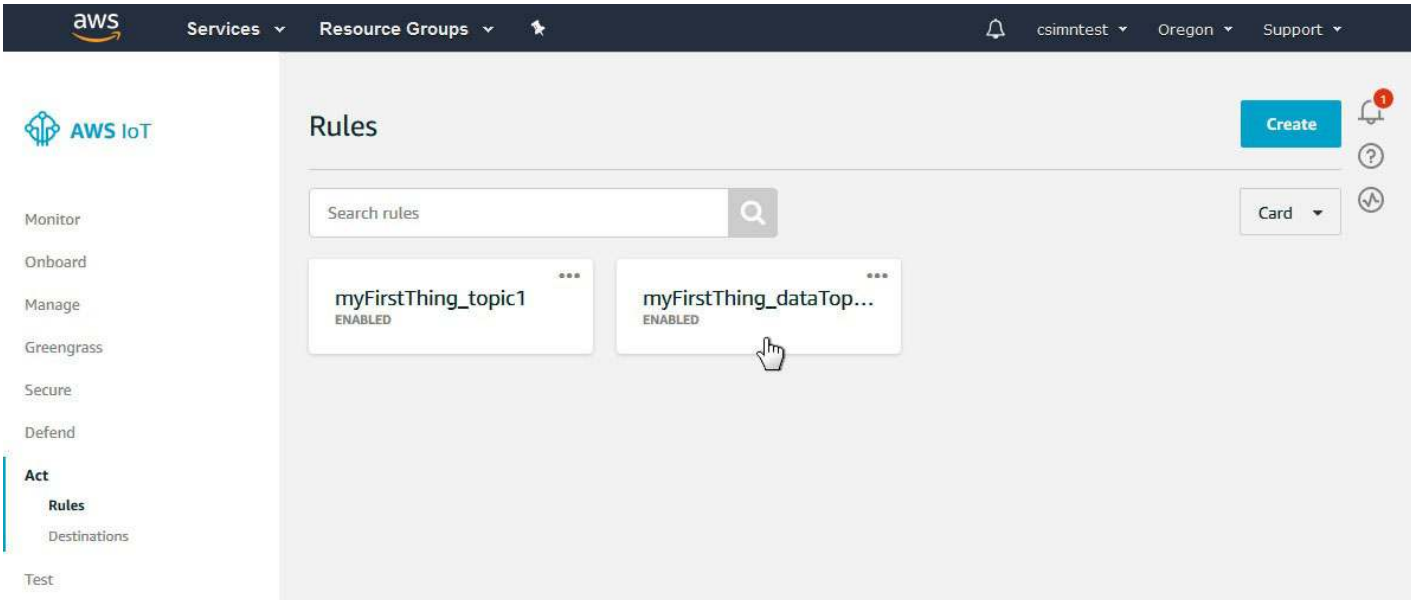
Set one or more actions

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. (*.required)

Send a message to IoT Analytics
myfirstthing_channel



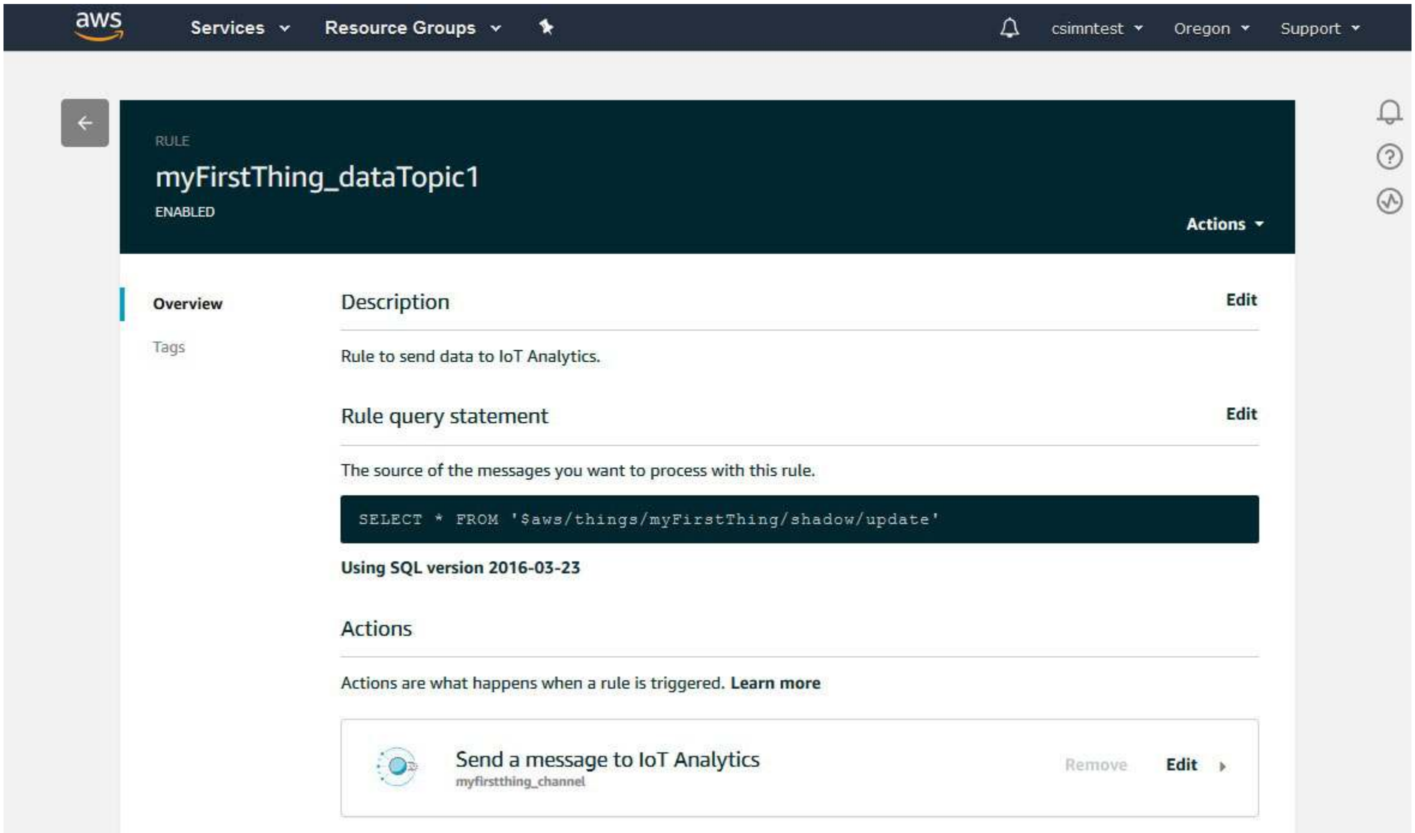
Your Rules page will now have a new rule. This illustration now includes rules from both Sections 16 and 17.



myFirstThing_topic1
ENABLED

myFirstThing_dataTop...
ENABLED

If you click on the rule at this point, it will be displayed as follows.



RULE

myFirstThing_dataTopic1

ENABLED

Description

Rule to send data to IoT Analytics.

Rule query statement

The source of the messages you want to process with this rule.

```
SELECT * FROM '$aws/things/myFirstThing/shadow/update'
```

Using SQL version 2016-03-23

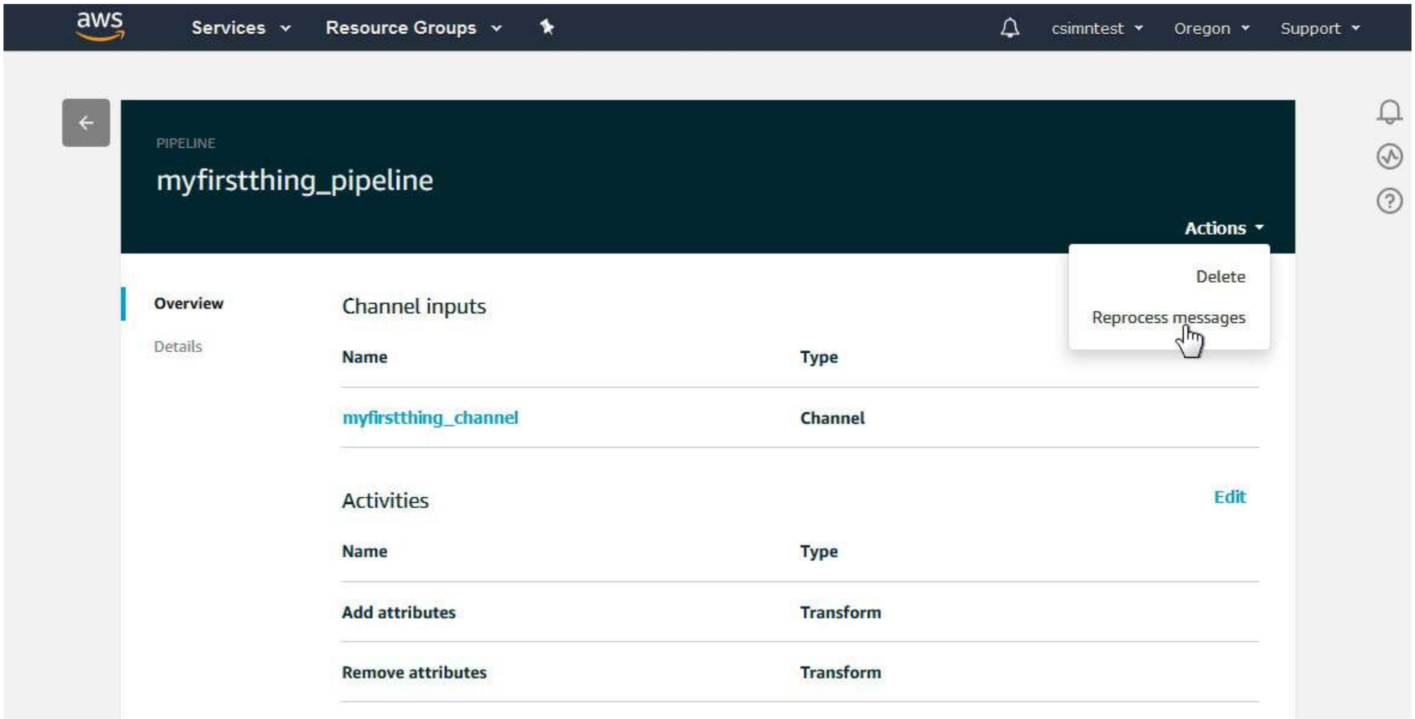
Actions

Actions are what happens when a rule is triggered. Learn more

Send a message to IoT Analytics
myfirstthing_channel

17.5 Process Data

You are now ready to process data. You will need to start by giving your Thing some time to publish data. Once you are confident there is at least some data to look at, you can come back to the pipeline and click Reprocess messages. This will go back and capture any data that might have gotten published while you were setting up IoT Analytics.



myfirstthing_pipeline

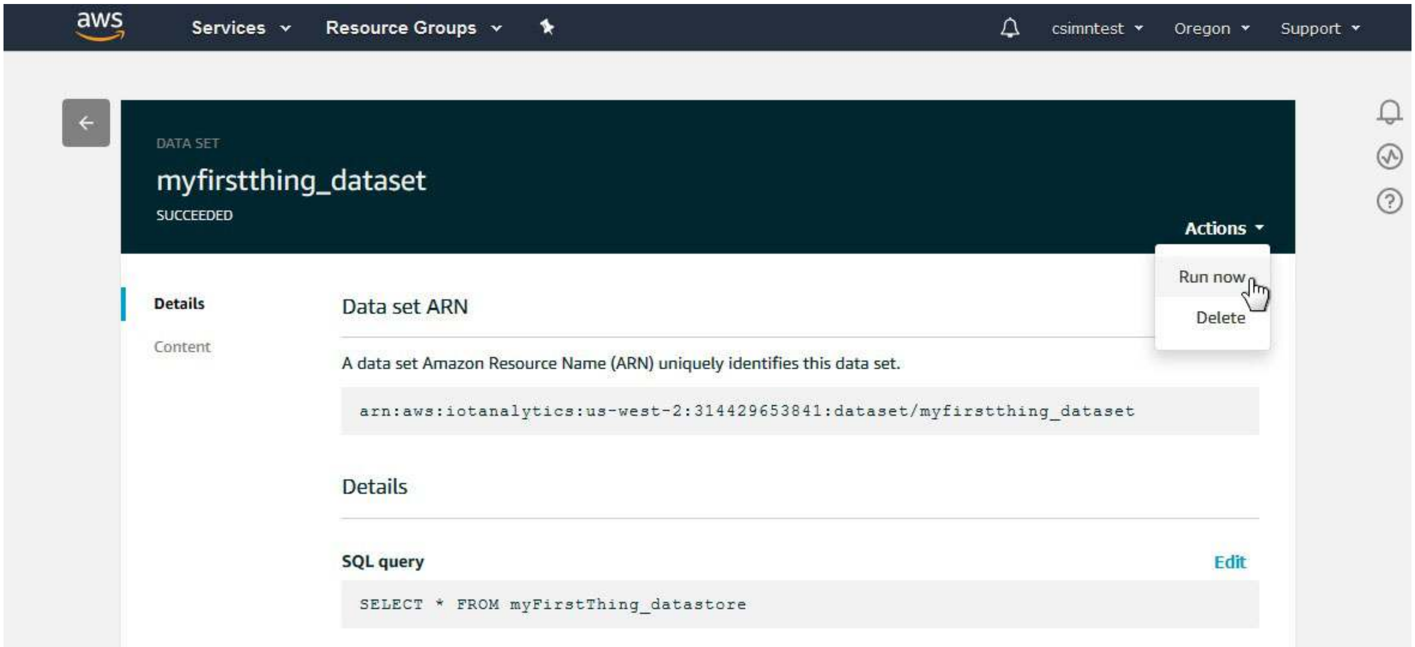
Channel inputs

Name	Type
myfirstthing_channel	Channel

Activities

Name	Type
Add attributes	Transform
Remove attributes	Transform

Go to the Data set and click Run now to rerun the SQL query.



myfirstthing_dataset

SUCCEEDED

Data set ARN

A data set Amazon Resource Name (ARN) uniquely identifies this data set.

```
arn:aws:iotanalytics:us-west-2:314429653841:dataset/myfirstthing_dataset
```

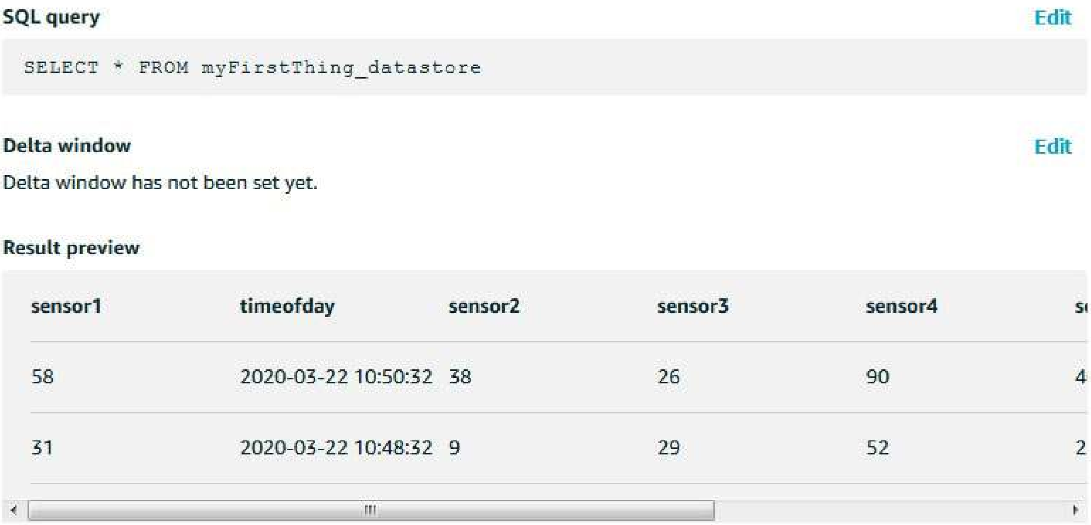
Details

SQL query

```
SELECT * FROM myFirstThing_datastore
```

The result will show as a preview on the Data set page. If you have set an update schedule (e.g. hourly), then you don't need to repeat the Run now when you come back to look some time later. Data will now be waiting for you any time you come back to look at it, and it will also be available for visualization using QuickSight.

It should be noted, however, that data collection in the data set is not promptly real time. There is some lag time between data being published and data reaching the data store. Notifications from SNS occur almost instantly. It has also been observed that data from the data set may be forwarded to QuickSight in real time but still not yet display in the query preview on this page.



SQL query

```
SELECT * FROM myFirstThing_datastore
```

Delta window

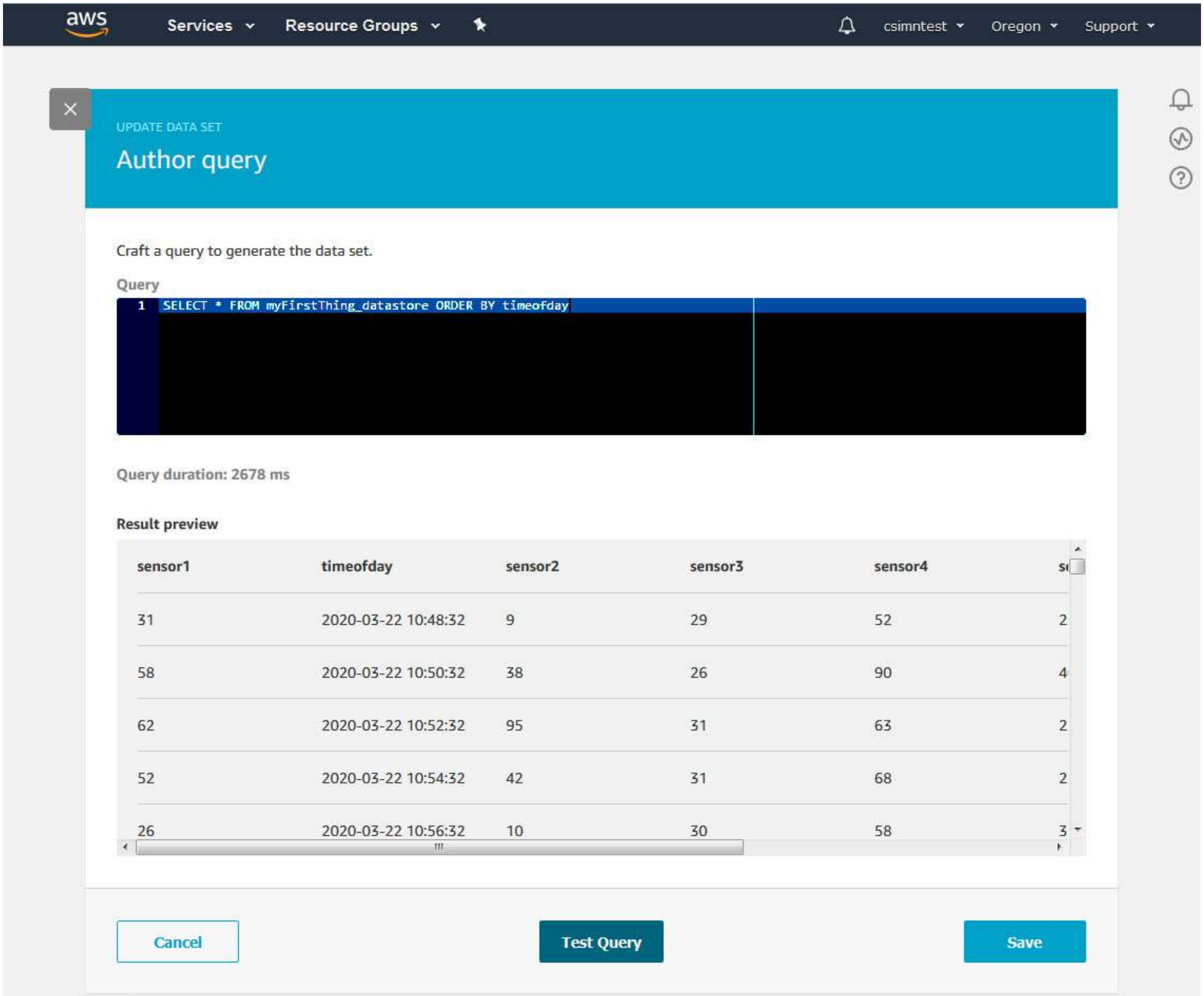
Delta window has not been set yet.

Result preview

sensor1	timeofday	sensor2	sensor3	sensor4	s
58	2020-03-22 10:50:32	38	26	90	4
31	2020-03-22 10:48:32	9	29	52	2

You have the option of modifying the SQL query to be anything that is a valid SQL query. You can replace 'SELECT *' with 'SELECT sensor1' for example. You can also add other SQL qualifiers. To modify the SQL query statement, click the Edit link next to SQL query.

Edit the query as you see fit. Test the query by clicking the Test Query button. If successful and there are no errors, click Save. The SQL edit window does not normally appear as bright as illustrated below - this screen shot was enhanced for readability in this document.



Author query

Craft a query to generate the data set.

Query

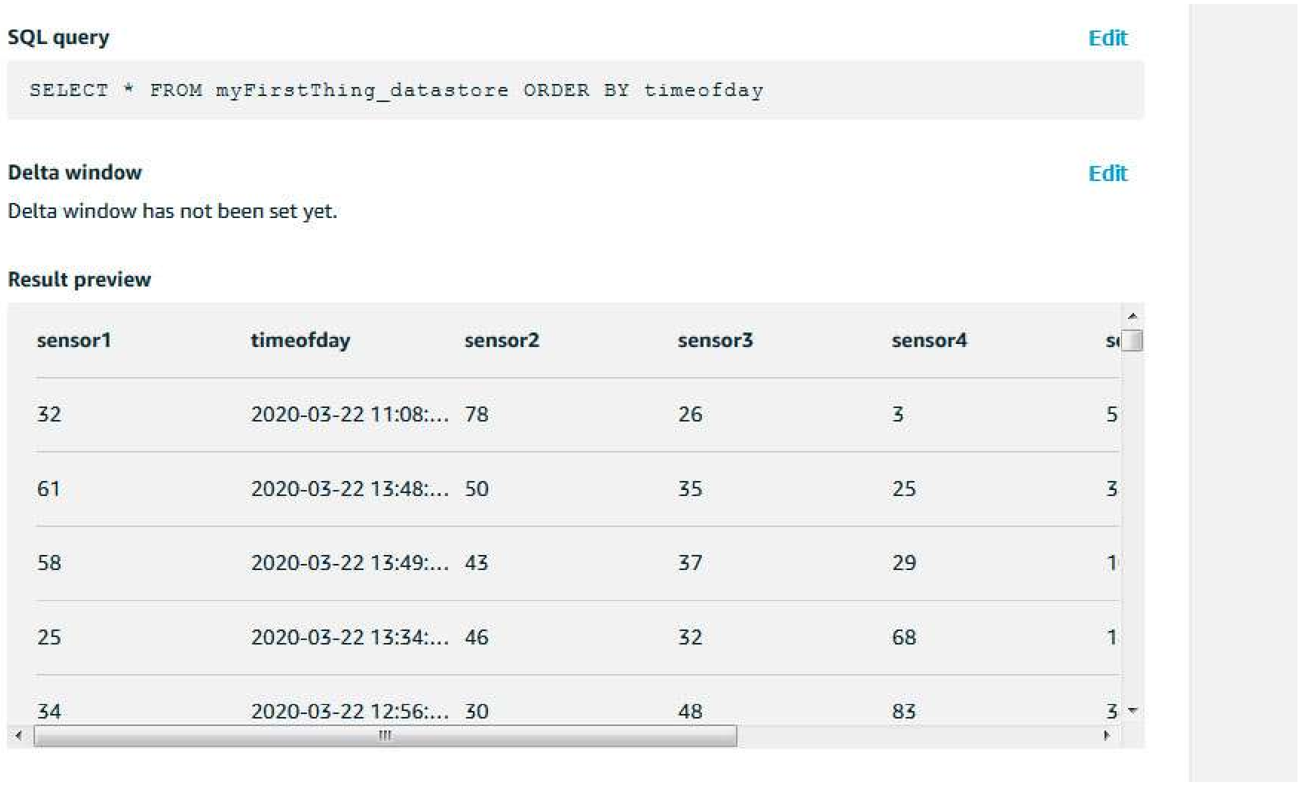
```
SELECT * FROM myFirstThing_datastore ORDER BY timeofday
```

Query duration: 2678 ms

Result preview

sensor1	timeofday	sensor2	sensor3	sensor4	s
31	2020-03-22 10:48:32	9	29	52	2
58	2020-03-22 10:50:32	38	26	90	4
62	2020-03-22 10:52:32	95	31	63	2
52	2020-03-22 10:54:32	42	31	68	2
26	2020-03-22 10:56:32	10	30	58	3

Once your SQL query edit is complete, the updated query will appear in the query window.



SQL query

```
SELECT * FROM myFirstThing_datastore ORDER BY timeofday
```

Delta window

Delta window has not been set yet.

Result preview

sensor1	timeofday	sensor2	sensor3	sensor4	s
32	2020-03-22 11:08:...	78	26	3	5
61	2020-03-22 13:48:...	50	35	25	3
58	2020-03-22 13:49:...	43	37	29	1
25	2020-03-22 13:34:...	46	32	68	1
34	2020-03-22 12:56:...	30	48	83	3

Babel Buster IoT
MODEL MQ-61
MODEL BB3-6101-MQ
IOT GATEWAY

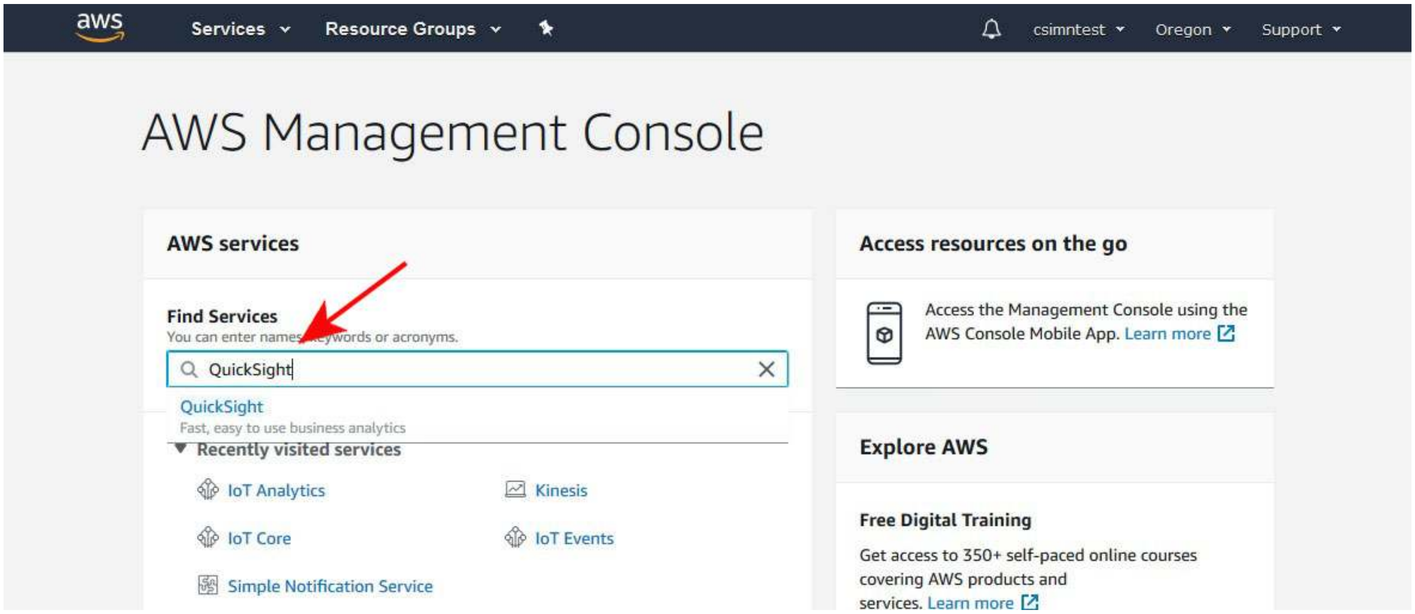
CONTROL SOLUTIONS MINNESOTA

18. Configuring QuickSight

AWS QuickSight provides a convenient means of quickly visualizing the data collected from your Thing.

18.1 Create New Analysis

Start by searching for QuickSight in your AWS Management Console.



AWS Management Console

AWS services

Find Services
You can enter names, keywords or acronyms.

QuickSight

QuickSight
Fast, easy to use business analytics

Recently visited services

IoT Analytics
Kinesis
IoT Core
IoT Events
Simple Notification Service

Access resources on the go

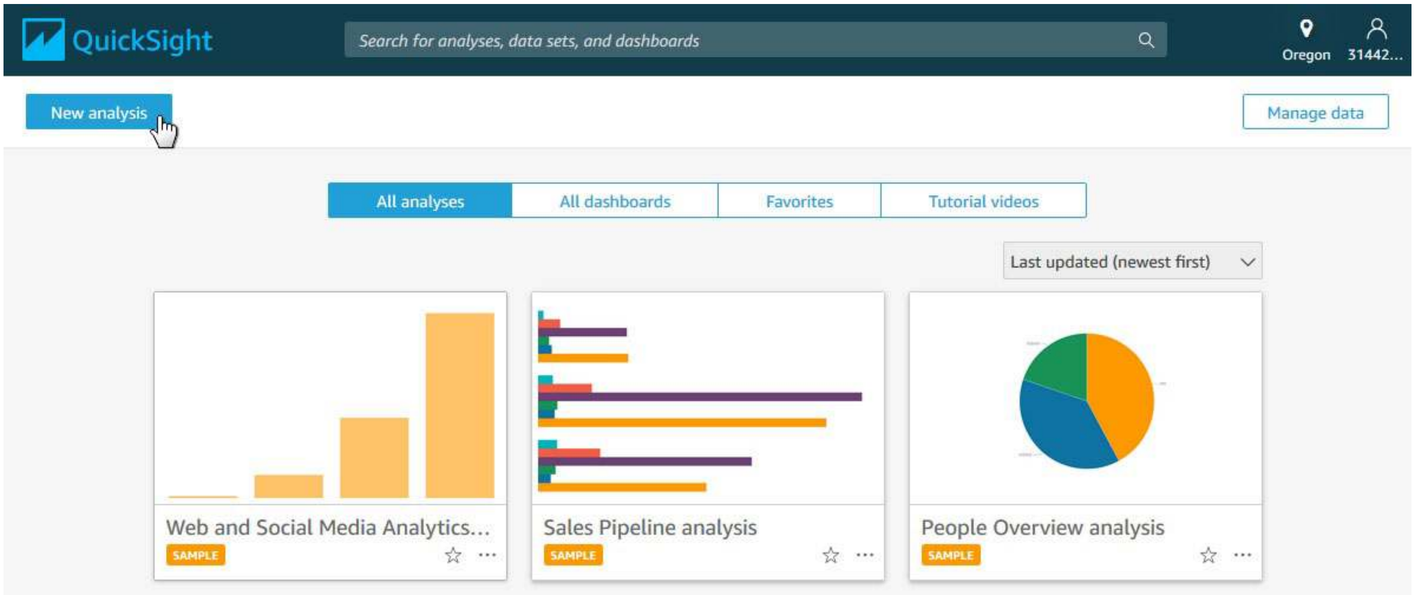
Access the Management Console using the AWS Console Mobile App. Learn more

Explore AWS

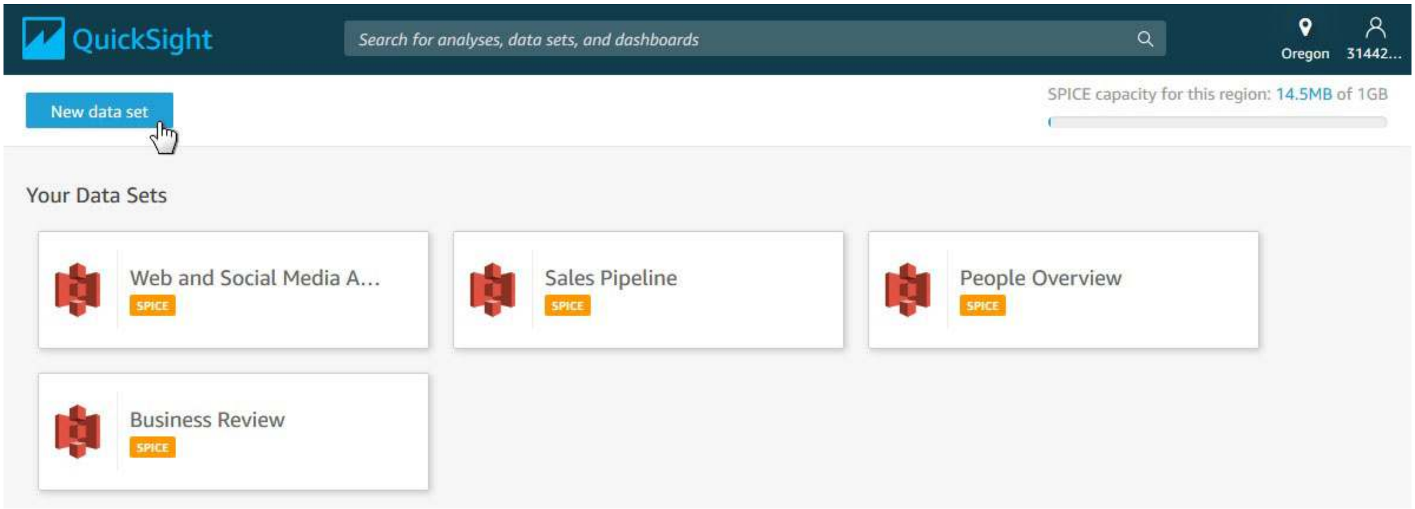
Free Digital Training

Get access to 350+ self-paced online courses covering AWS products and services. Learn more

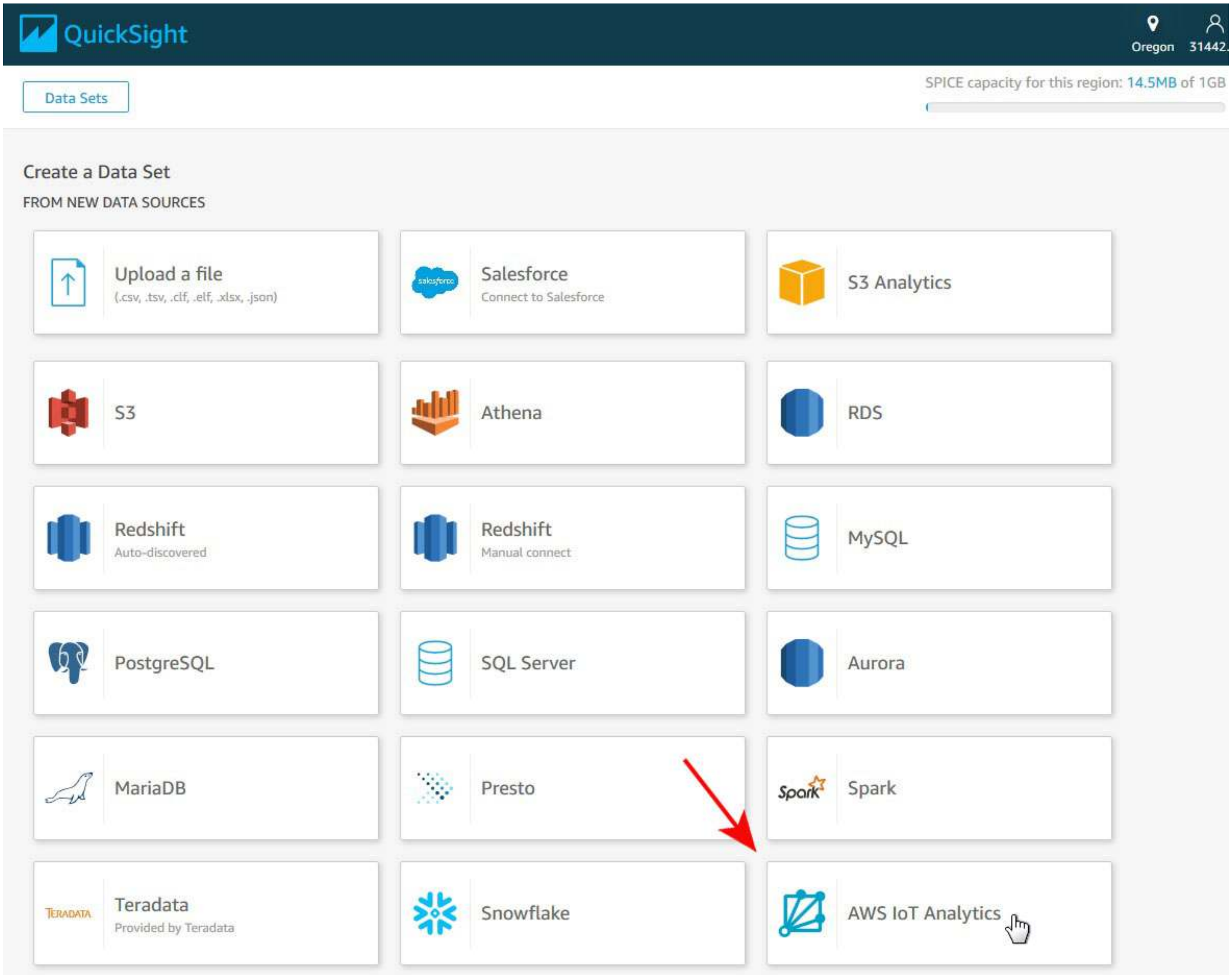
The QuickSight dashboard is an entirely different site independent of the AWS Management Console. Upon first visit, only some example visualizations are provided on the dashboard. Click "New analysis".



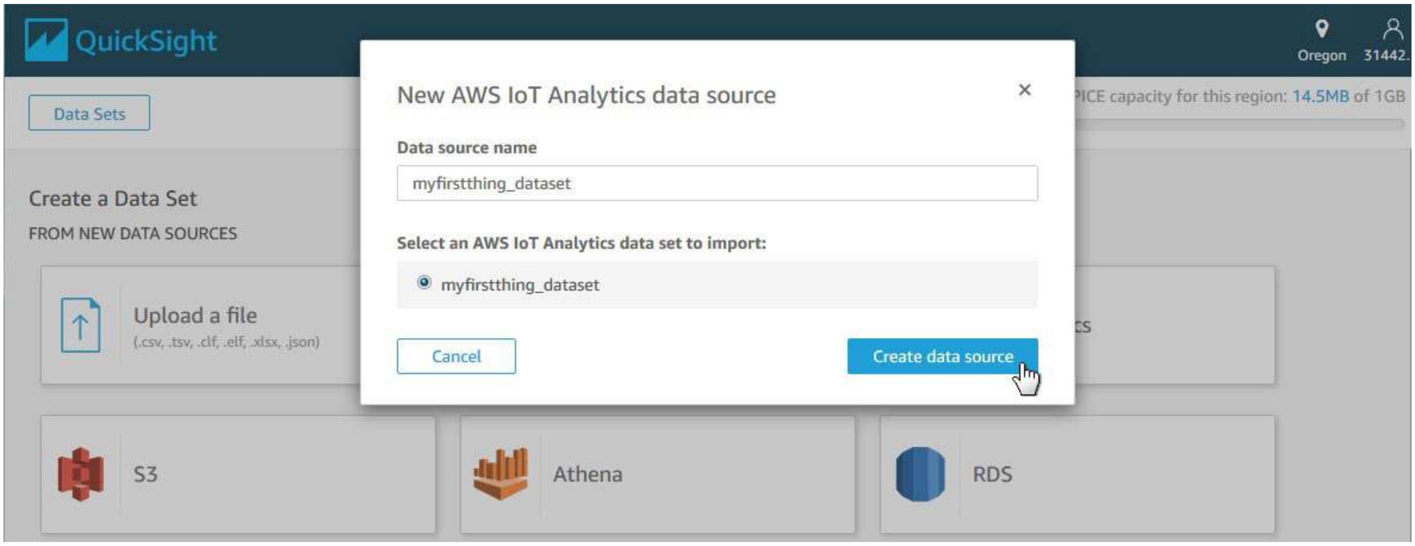
The first step in creating an analysis is to select a data set. Click "New data set".


QuickSight
Search for analyses, data sets, and dashboards
Oregon
31442...
New data set
SPICE capacity for this region: 14.5MB of 1GB
Your Data Sets
Web and Social Media A...
SPICE
Sales Pipeline
SPICE
People Overview
SPICE
Business Review
SPICE

You have a long list of choices for sources of your data set. Find AWS IoT Analytics and select it.



The data set you created in IoT Analytics will be displayed. Select it, provide a name (will appear on QuickSight dashboard) and click Create data source.



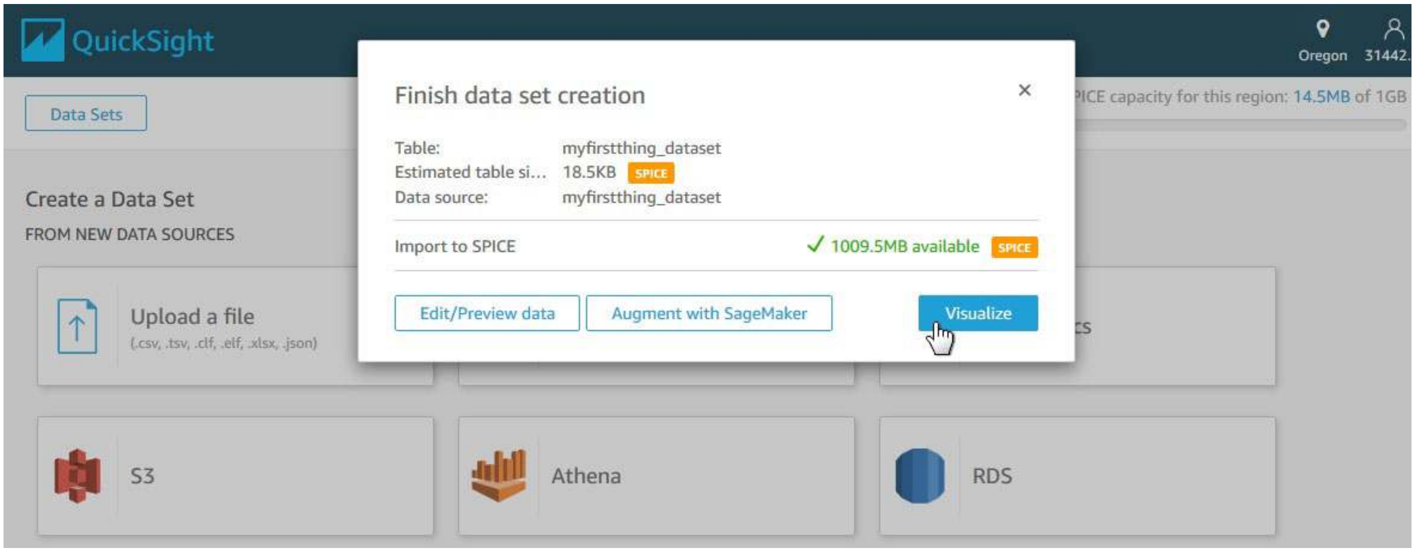
New AWS IoT Analytics data source

Data source name: myfirstthing_dataset

Select an AWS IoT Analytics data set to import:

● myfirstthing_dataset

Upon clicking "Create data source", the display will appear as follows. Now click Visualize.



Finish data set creation

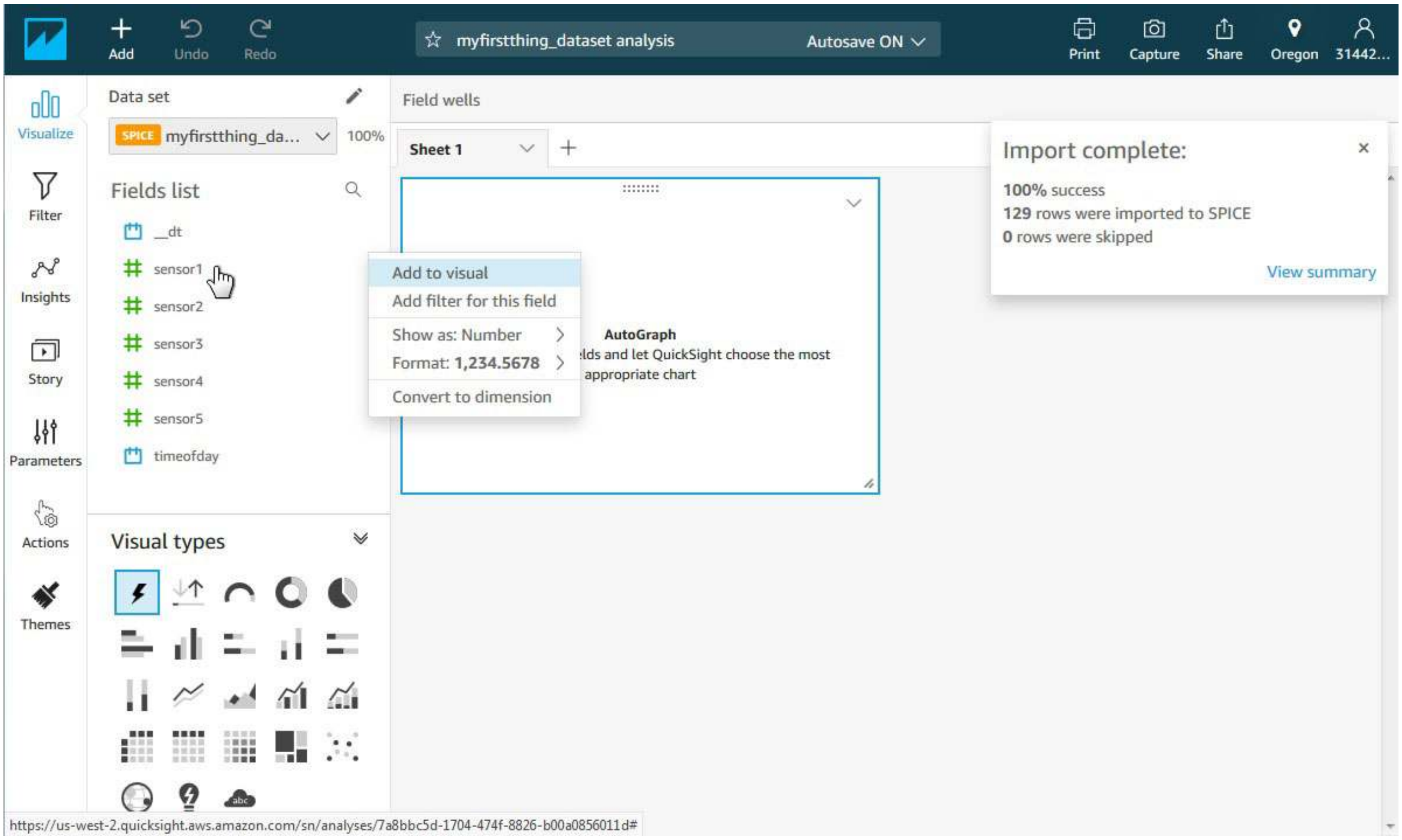
Table: myfirstthing_dataset
Estimated table si... 18.5KB SPICE
Data source: myfirstthing_dataset

Import to SPICE ✓ 1009.5MB available SPICE

PICE capacity for this region: 14.5MB of 1GB

18.2 Visualize Data

The initial visualize screen will appear as shown below. Click on fields on the fields list and select Add to visual.

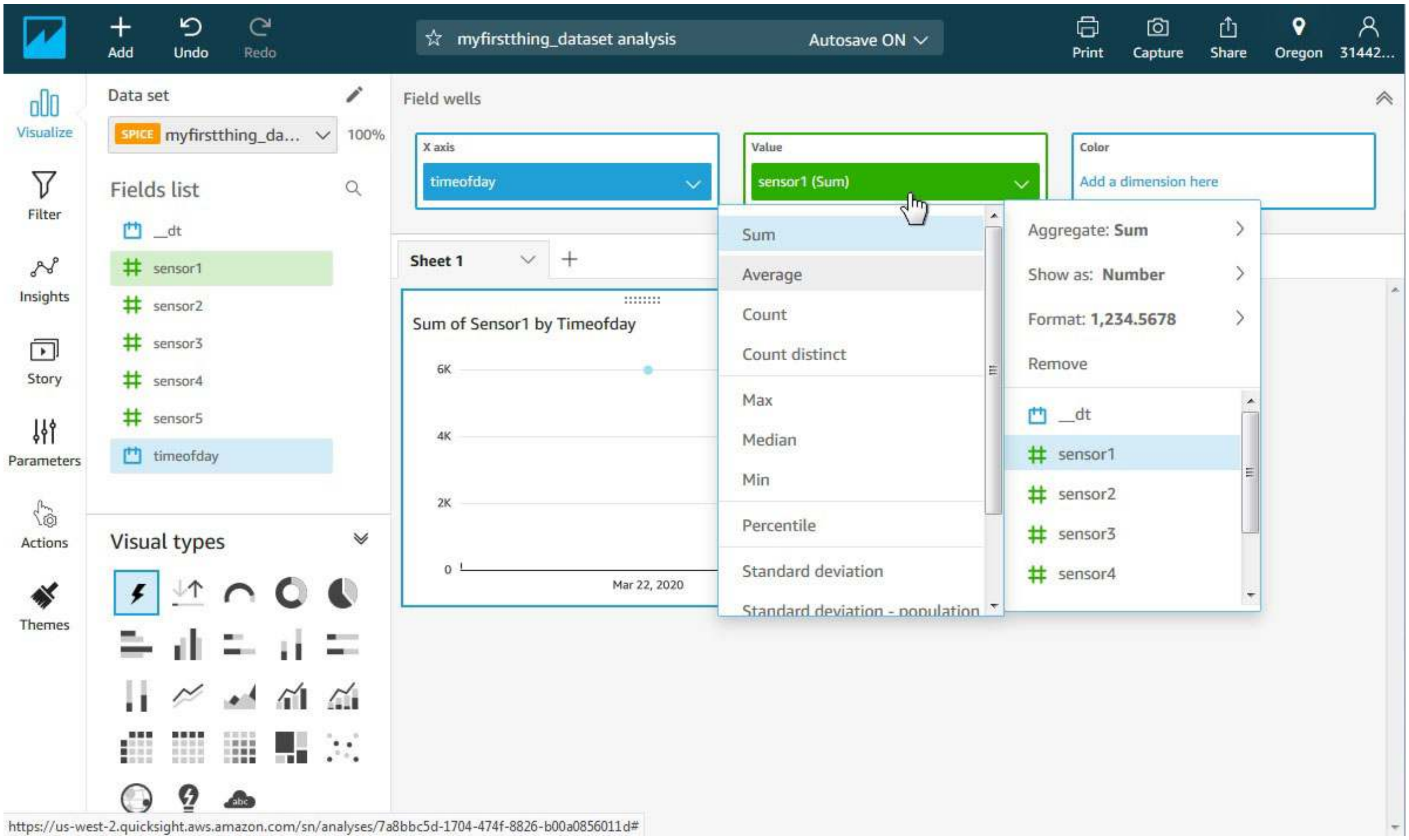


Import complete:

100% success
129 rows were imported to SPICE
0 rows were skipped

https://us-west-2.quicksight.aws.amazon.com/sn/analyses/7a8bbc5d-1704-474f-8826-b00a0856011d#

For our example, we are using timeofday as the X axis, and the sensor values on the Y axis. Once the sensor is added to the Value list, you can further select optional manipulations such as Sum or Average (per time period).



X axis: timeofday

Value: sensor1 (Sum)

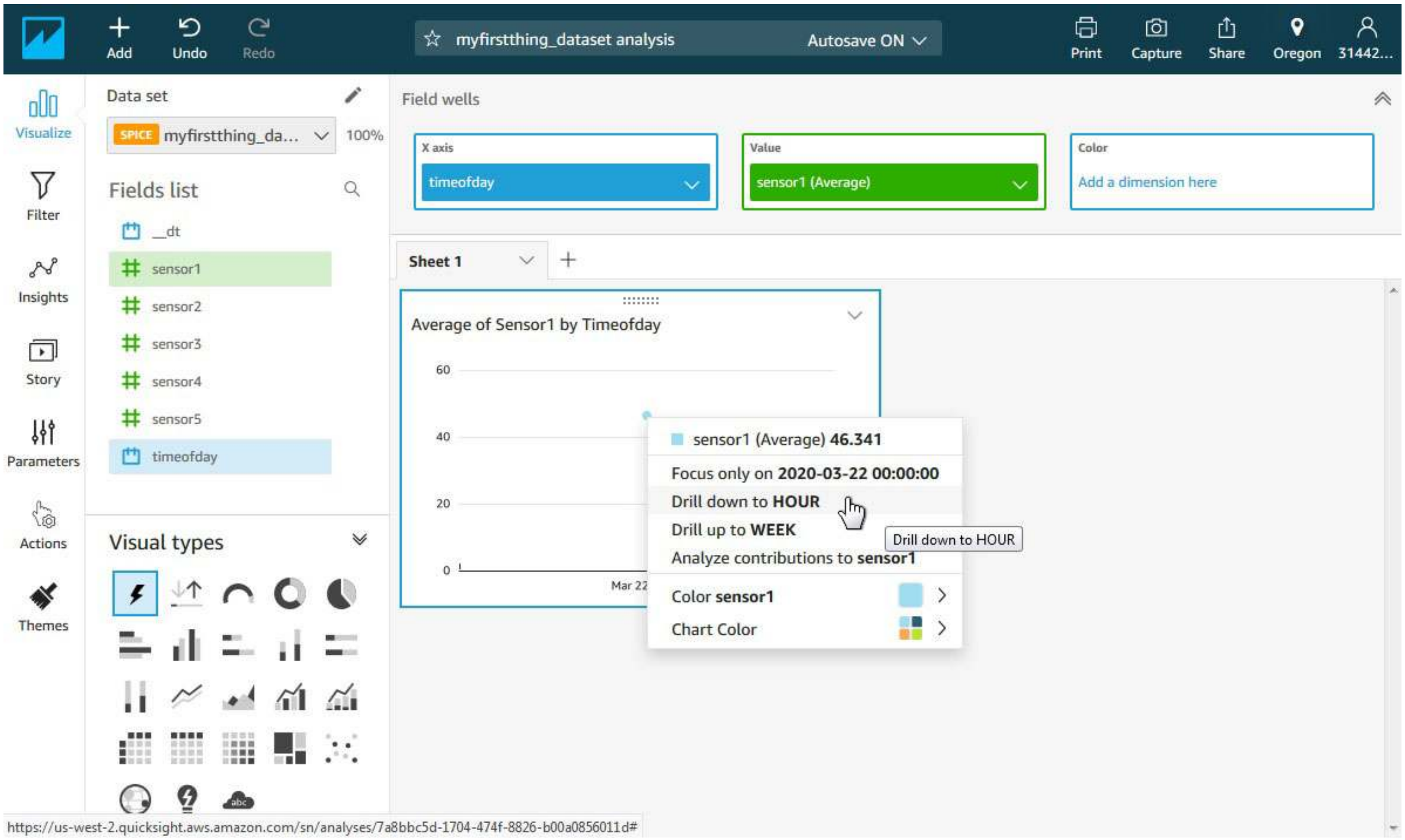
Color: Add a dimension here

Aggregate: Sum

Show as: Number

Format: 1,234.5678

The first time you attempt to visualize data, you might not have data spanning multiple days yet. If that is the case, place the mouse over the lone data point, and select Drill down to HOUR.



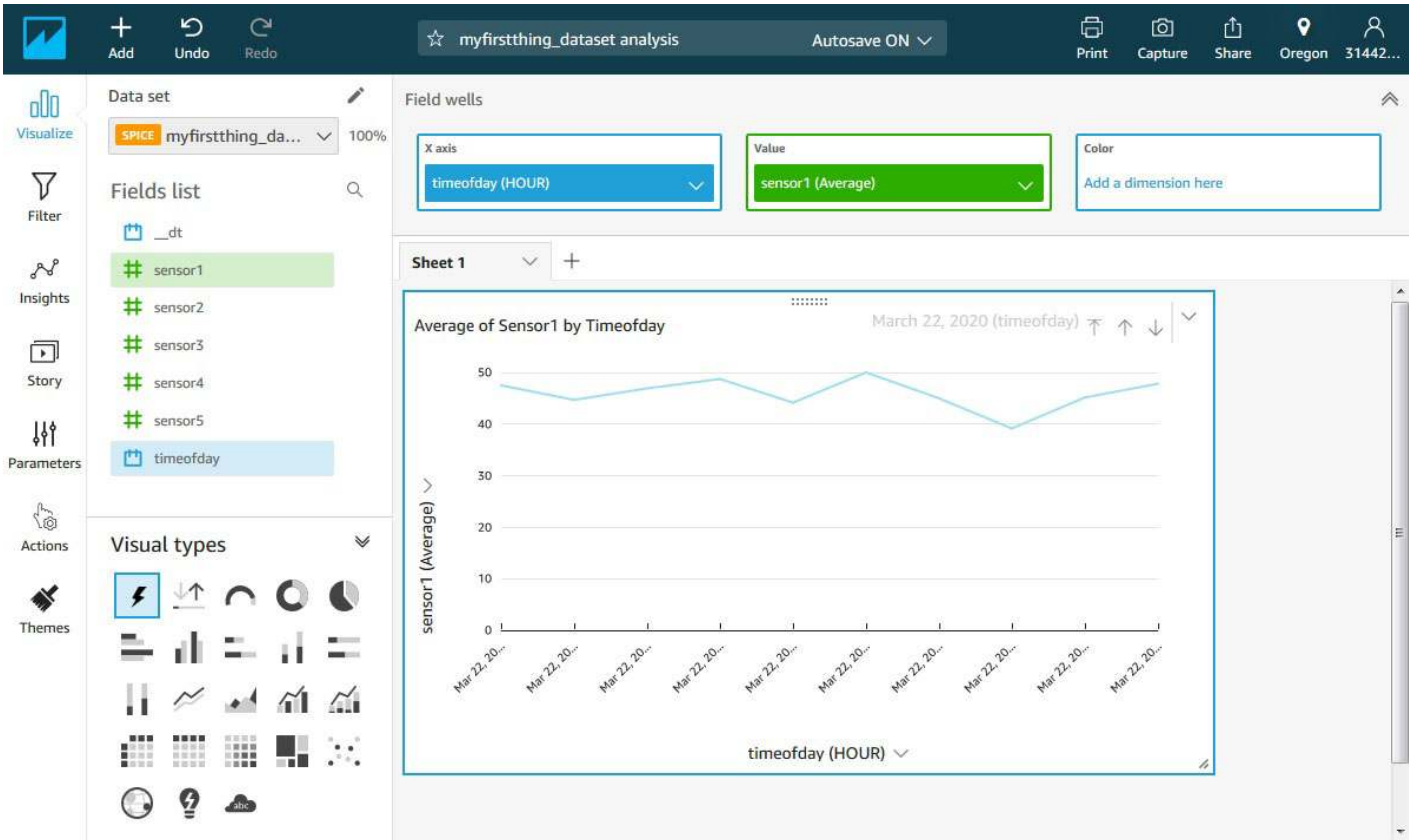
X axis: timeofday

Value: sensor1 (Average)

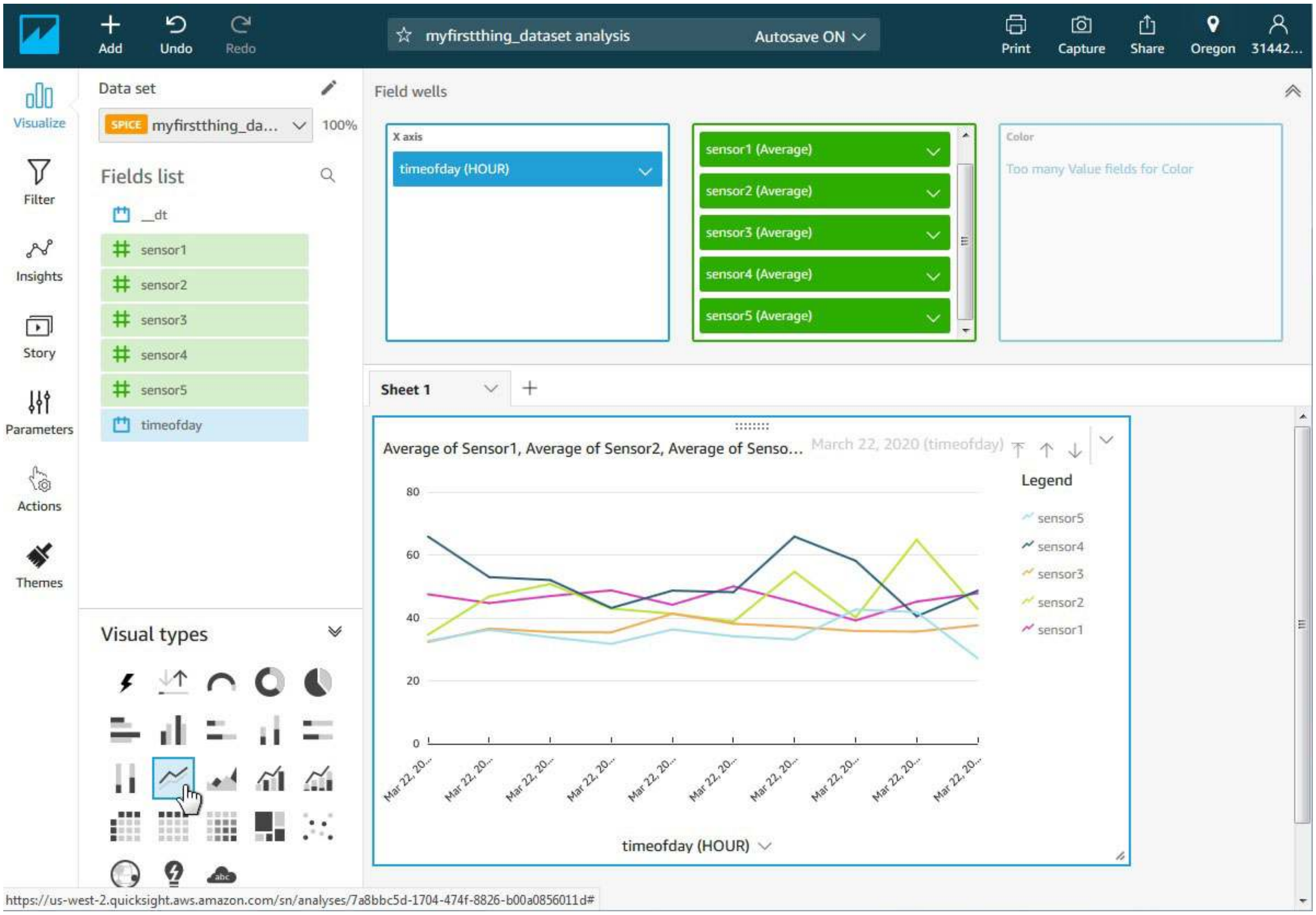
Color: Add a dimension here

sensor1 (Average) 46.341

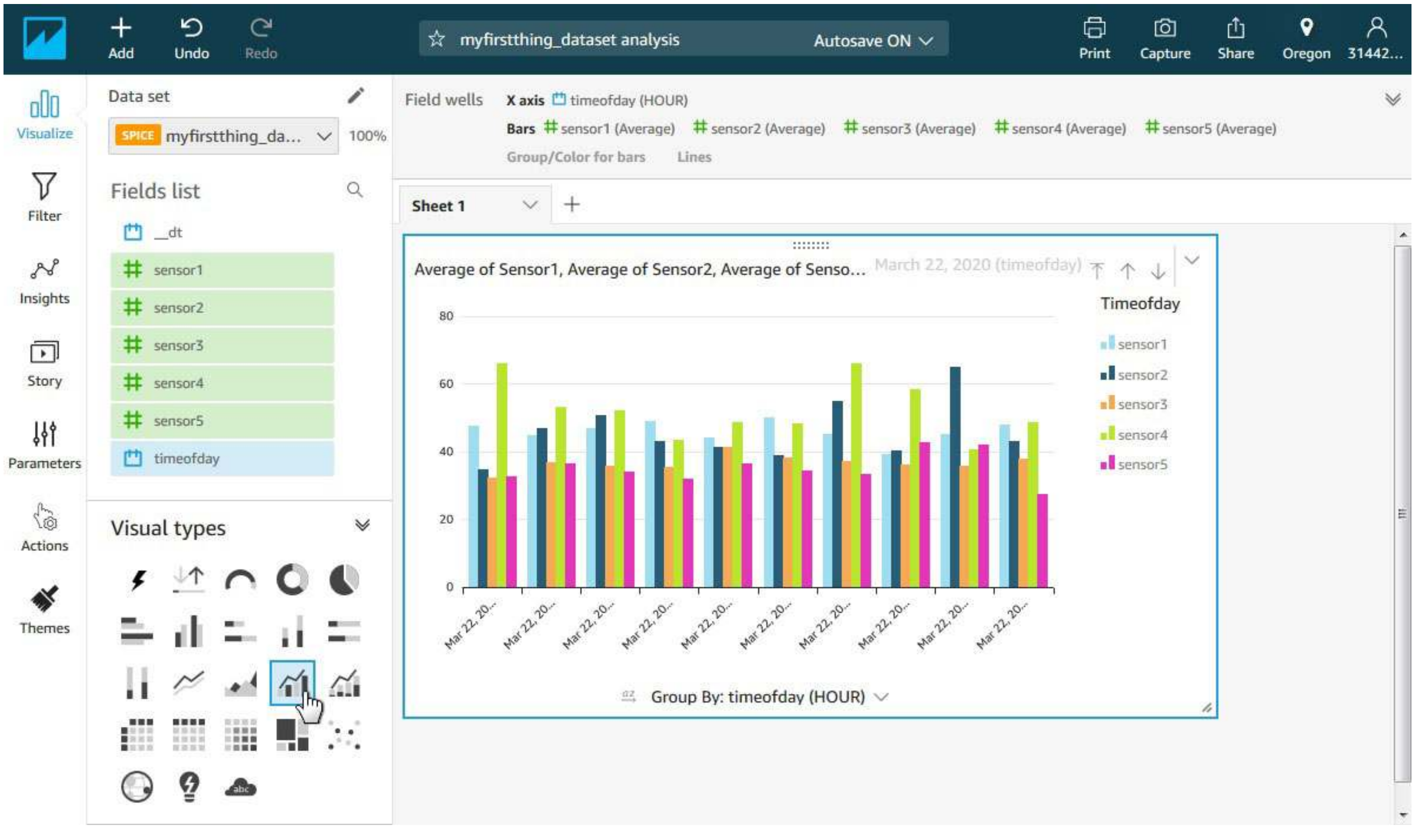
The resulting graph displayed so far looks like this:



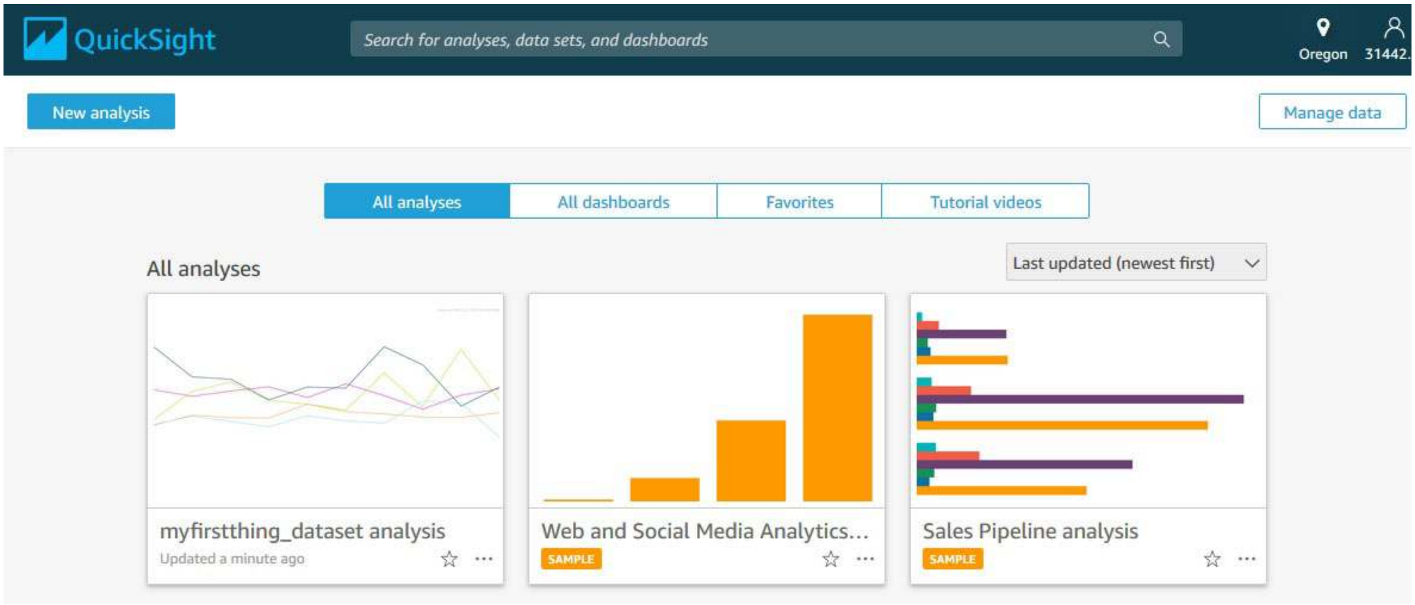
We repeat the process of adding values to add the rest of the sensors. Now we have the visualization illustrated below.



There are a number of Visual types you may choose from. Simple click on a new Visual type to change the format of the display. Be aware that some types of visualization only apply to a single data point, and selecting them will remove all but one of your values from the display list.



When you exit the visualization screen by clicking the QuickSight icon in the top left corner, you return to the dashboard where your new visualization is now displayed. The next time you return to QuickSight, simply click on your newly added dashboard icon to return to it.



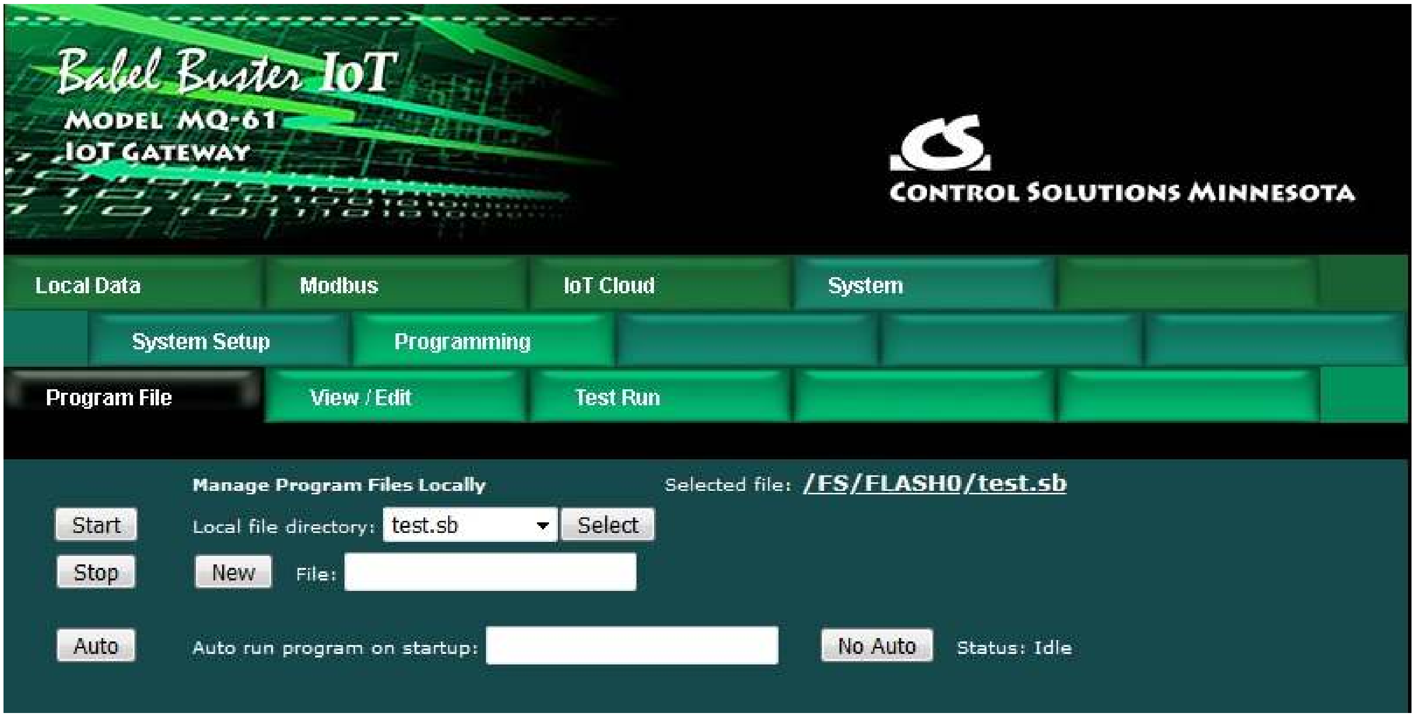
Babel Buster IoT
MODEL MQ-61
MODEL BB3-6101-MQ
IOT GATEWAY
CONTROL SOLUTIONS MINNESOTA

19. Programming with Script Basic

19.1 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 Babel Buster IoT Gateway includes a very simple text editing page under the View/Edit tab, and this is suitable for creating small programs or making minor changes.

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.



Manage Program Files Locally

Selected file: /FS/FLASH0/test.sb

Local file directory: test.sb

File:

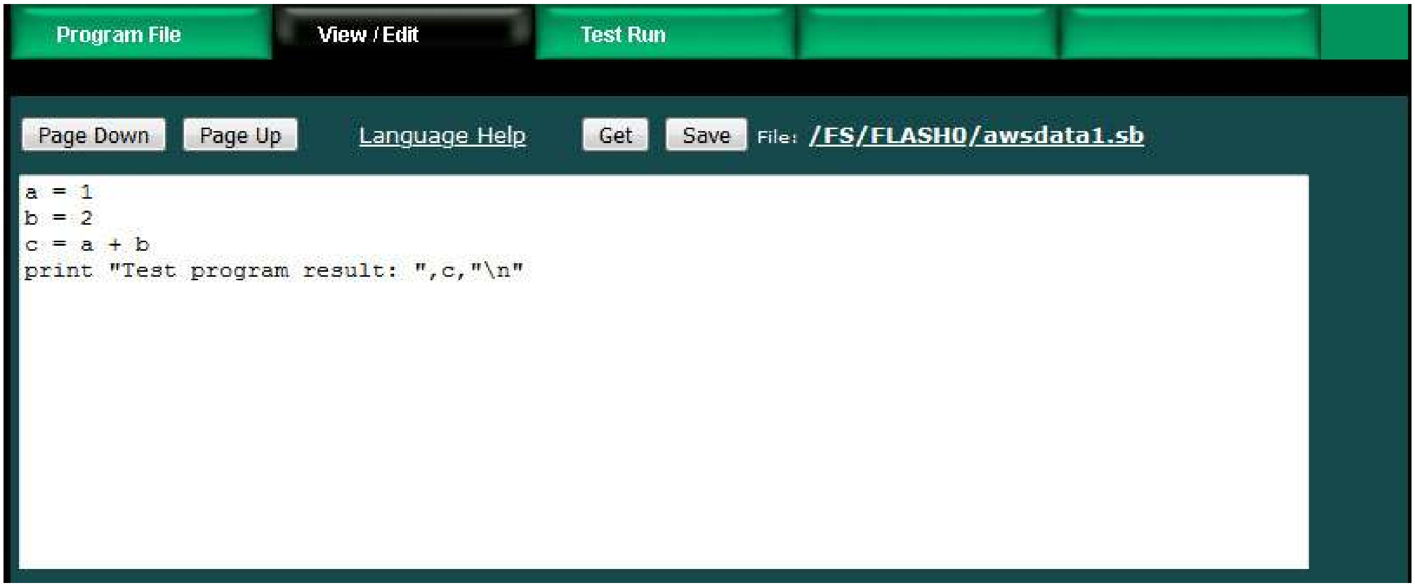
Auto run program on startup:

Status: Idle

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.



```
a = 1
b = 2
c = a + b
print "Test program result: ",c,"\n"
```

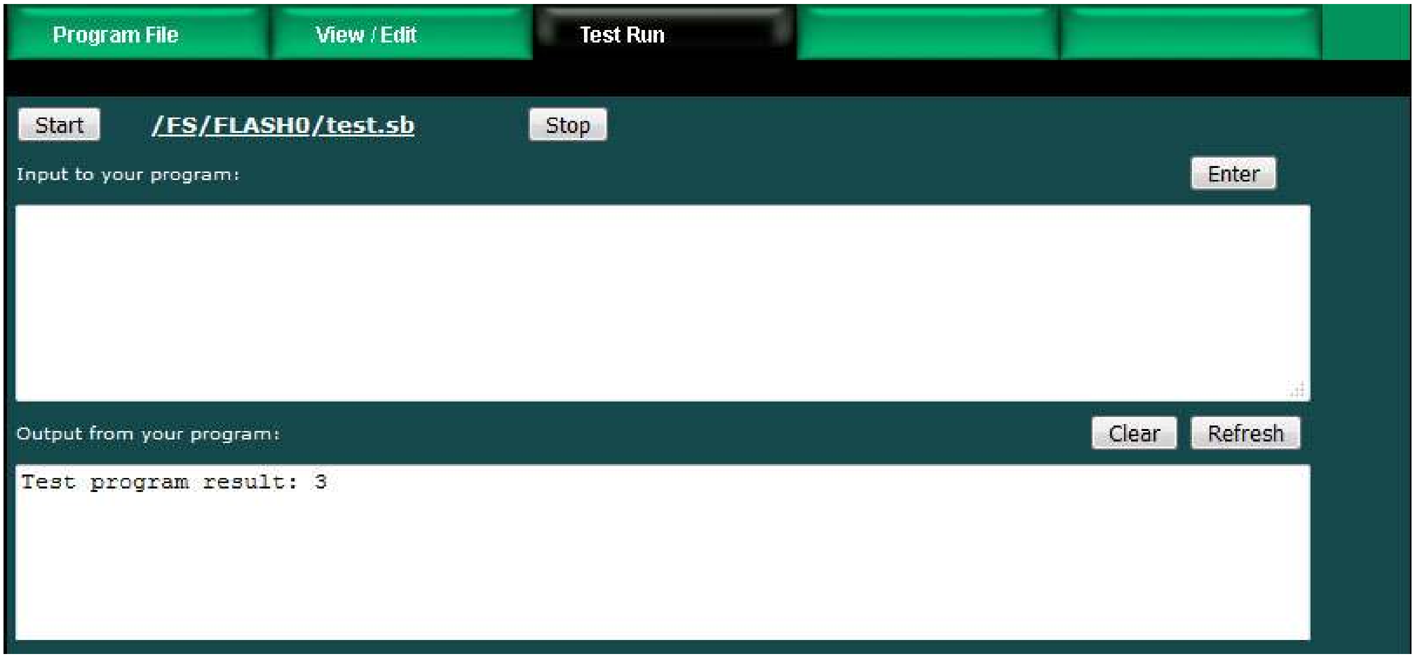
19.2 Testing the Program

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 Babel Buster. 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.

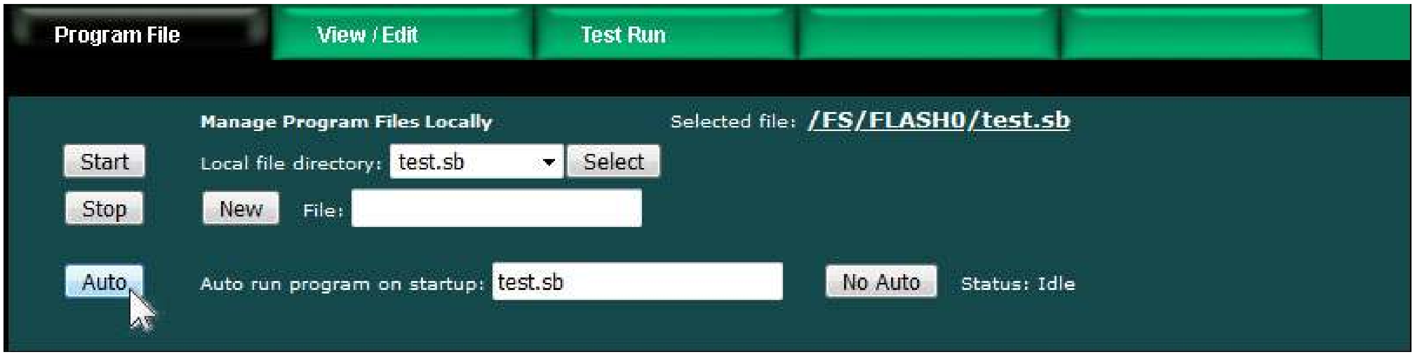


```
Test program result: 3
```

19.3 Setting the Program to Auto-Run on Startup

Your program will not be very useful if it does not start up when the Babel Buster IoT Gateway 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".



Selected file: /FS/FLASH0/test.sb
Local file directory: test.sb
Auto run program on startup: test.sb
Status: Idle

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 Babel Buster IoT Gateway to halt any program that was already running from the last bootup.

19.4 Special Functions

19.4.1 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.

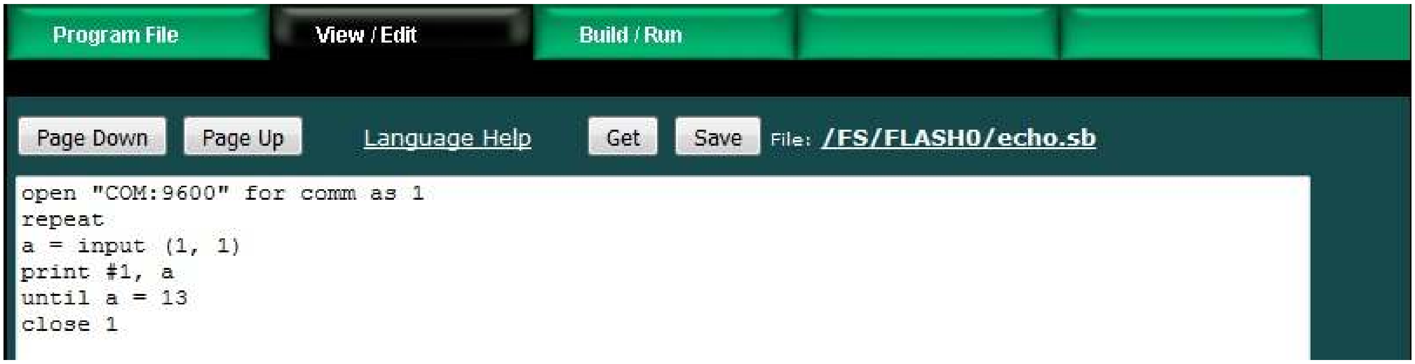
Input from Comm Port:

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.



```
open "COM:9600" for comm as 1
repeat
a = input (1, 1)
print #1, a
until a = 13
close 1
```

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 communiation 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.

19.4.2 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.

19.4.3 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.

19.4.4 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 successul, or 0 if it failed.

19.4.5 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.

19.4.6 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 withing this amount of time, the gateway will be restarted (as if power cycled).

19.4.7 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 gateway

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

19.4.8 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.

19.4.9 Free Memory

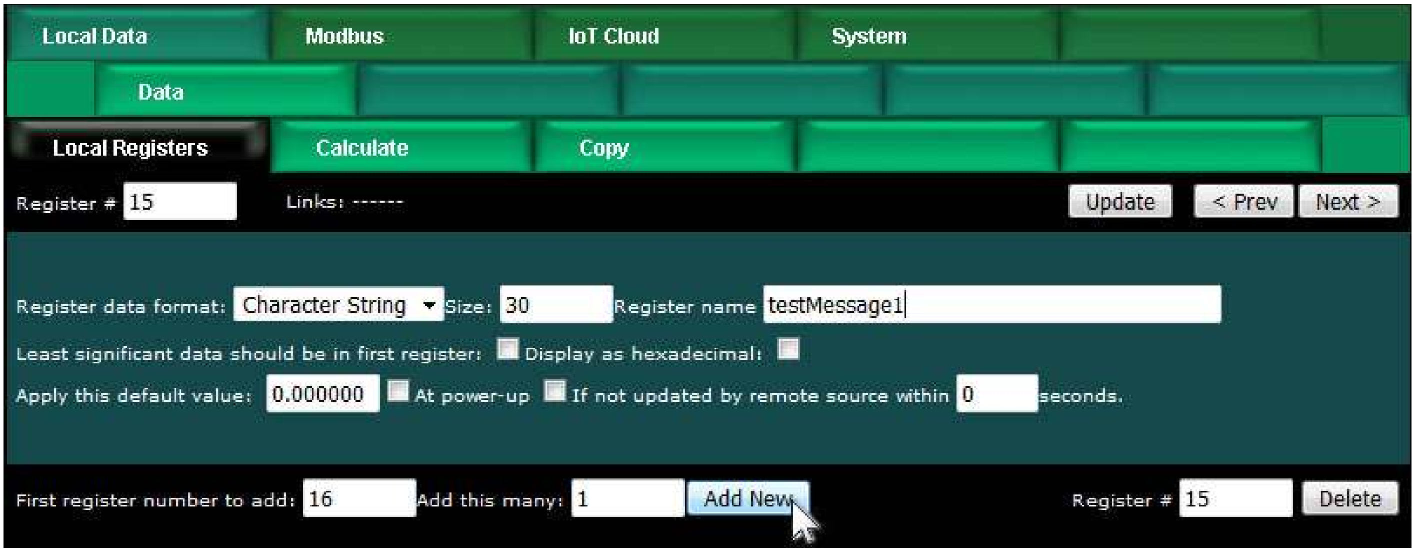
You can find out within your program how many bytes of free memory remain. In the line below, 'n' will be the number of bytes. It will typically start out over 1,500,000. When it drops to near zero, the system will stop functioning because memory allocation from the heap has run up against stack space.

```
    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.

19.5 Example: Program Triggering of Publish to AWS

The Babel Buster IoT Gateway 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.



Register # 15 Links: ------

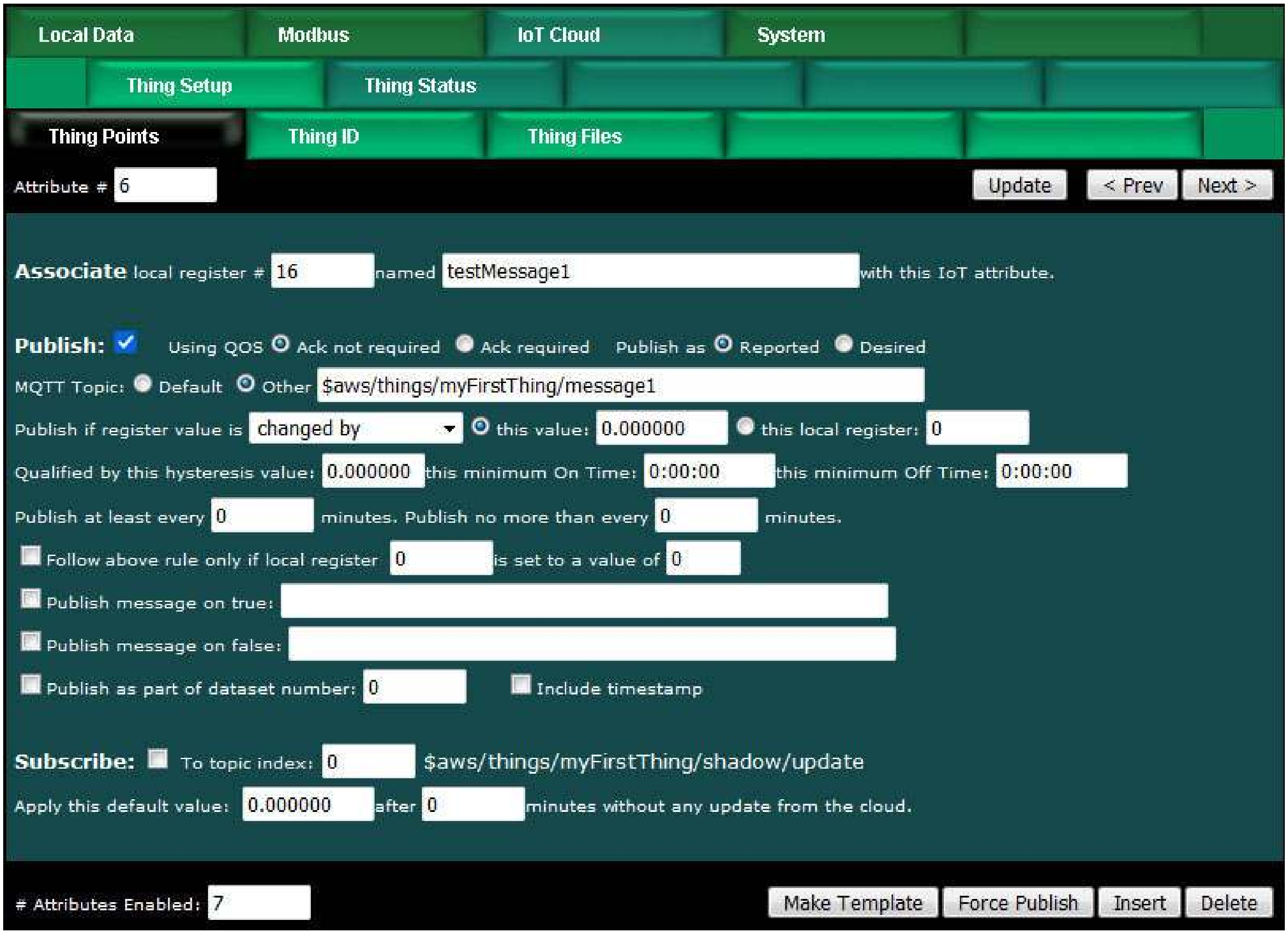
Register data format: Character String Size: 30 Register name testMessage1

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

Apply this default value: 0.000000 □ At power-up □ If not updated by remote source within 0 seconds.

First register number to add: 16 Add this many: 1 Register # 15

We start by creating a character string "register", which is really a series of 15 Modbus registers treated as one single data item.



Attribute # 6

Associate local register # 16 named testMessage1 with this IoT attribute.

Publish: ☑ Using QOS ● Ack not required ○ Ack required Publish as ● Reported ○ Desired

MQTT Topic: ○ Default ● Other $aws/things/myFirstThing/message1

Publish if register value is changed by ● 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:

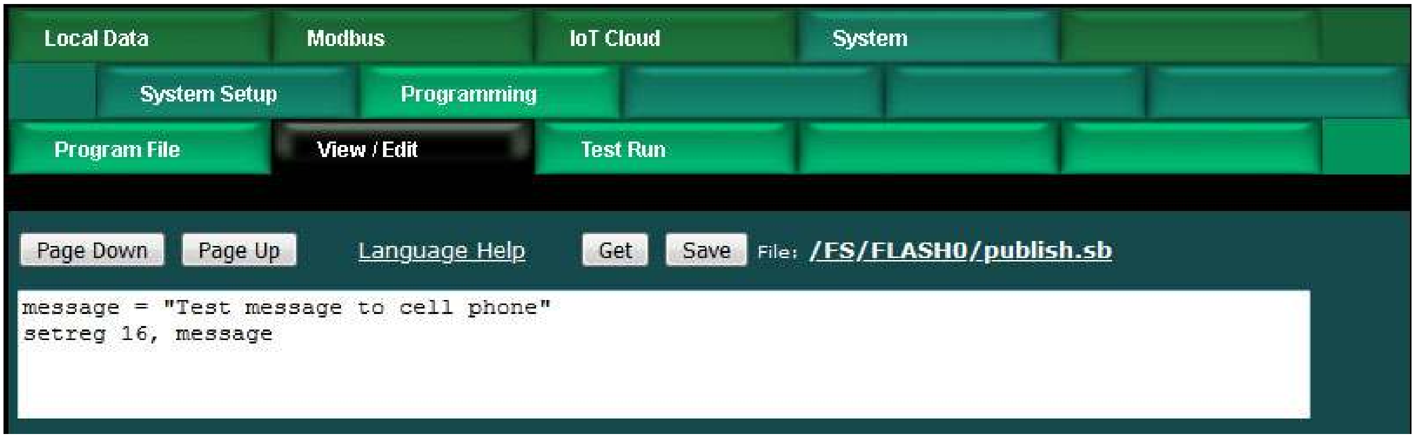
□ Publish as part of dataset number: 0 □ Include timestamp

Subscribe: □ To topic index: 0 $aws/things/myFirstThing/shadow/update

Apply this default value: 0.000000 after 0 minutes without any update from the cloud.

Attributes Enabled: 7

We have configured an IoT Thing Point (Attribute) to publish to the "other" topic that was set up in the Simple Notification Service section earlier in this user guide. By setting the rule to publish on "change by zero", we are telling the gateway to publish any update of the local register.

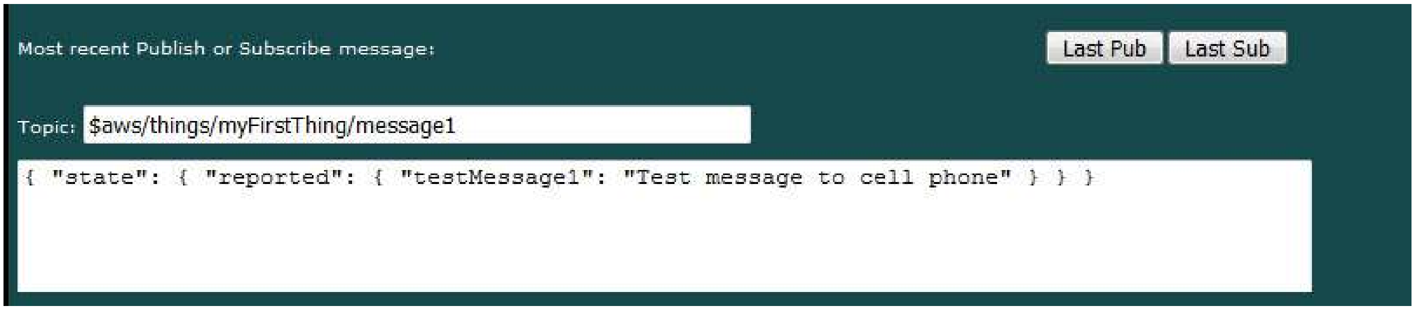


File: /FS/FLASH0/publish.sb

```
message = "Test message to cell phone"
setreg 16, message
```

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. This is just a variation on the theme first illustrated in Section 16, Configuring 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.

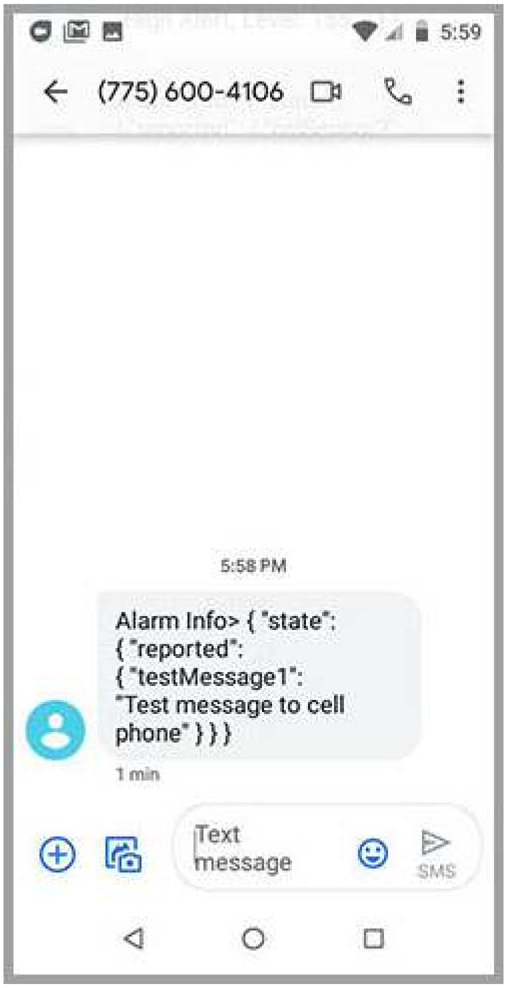


Most recent Publish or Subscribe message:

Topic: $aws/things/myFirstThing/message1

```
{ "state": { "reported": { "testMessage1": "Test message to cell phone" } } }
```

Upon running the test program, the publish message can be viewed on the Thing Status :: Test page to verify that it was sent.

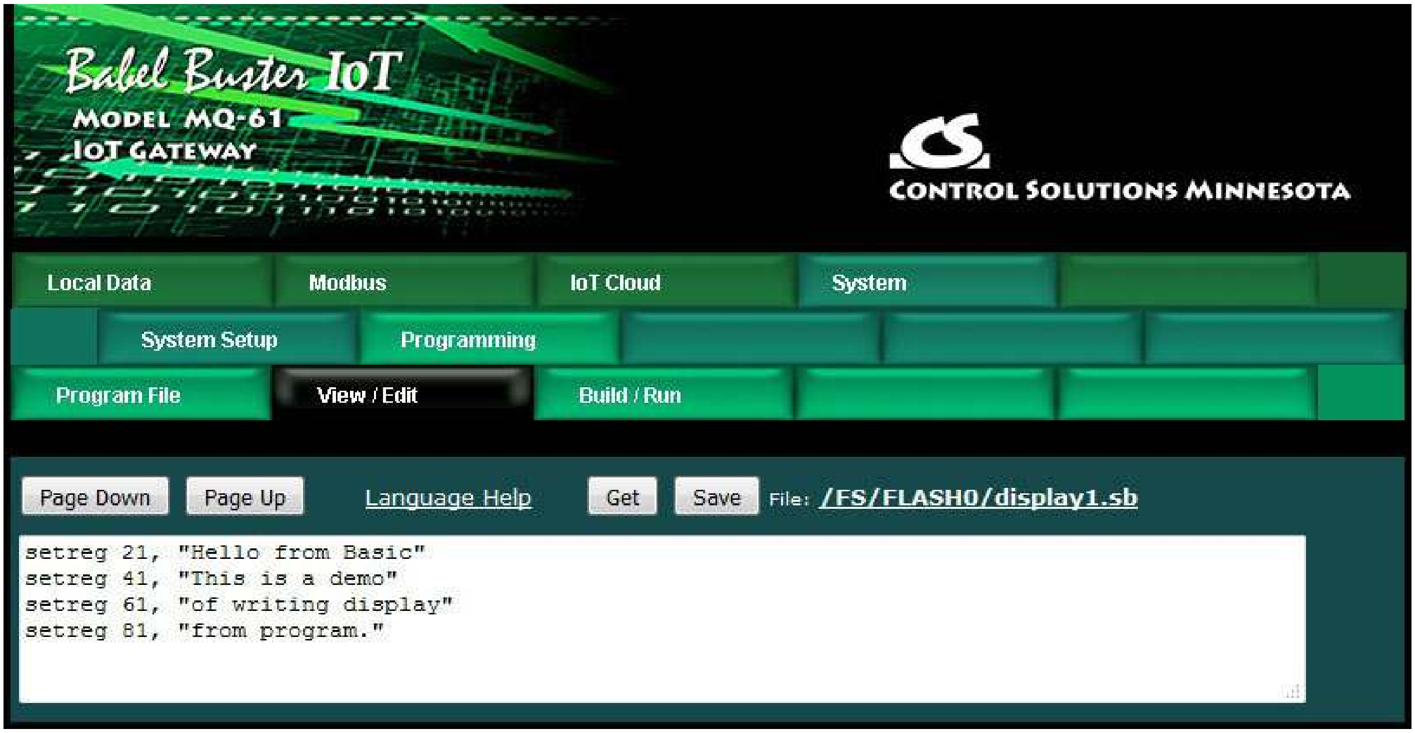


Alarm Info> { "state": { "reported": { "testMessage1": "Test message to cell phone" } } }

The combination of the Simple Notification Service set up in Section 16 and the simple test program illustrated here results in the text message being displayed on the mobile phone.

19.6 Example: Writing to an Alphanumeric Display

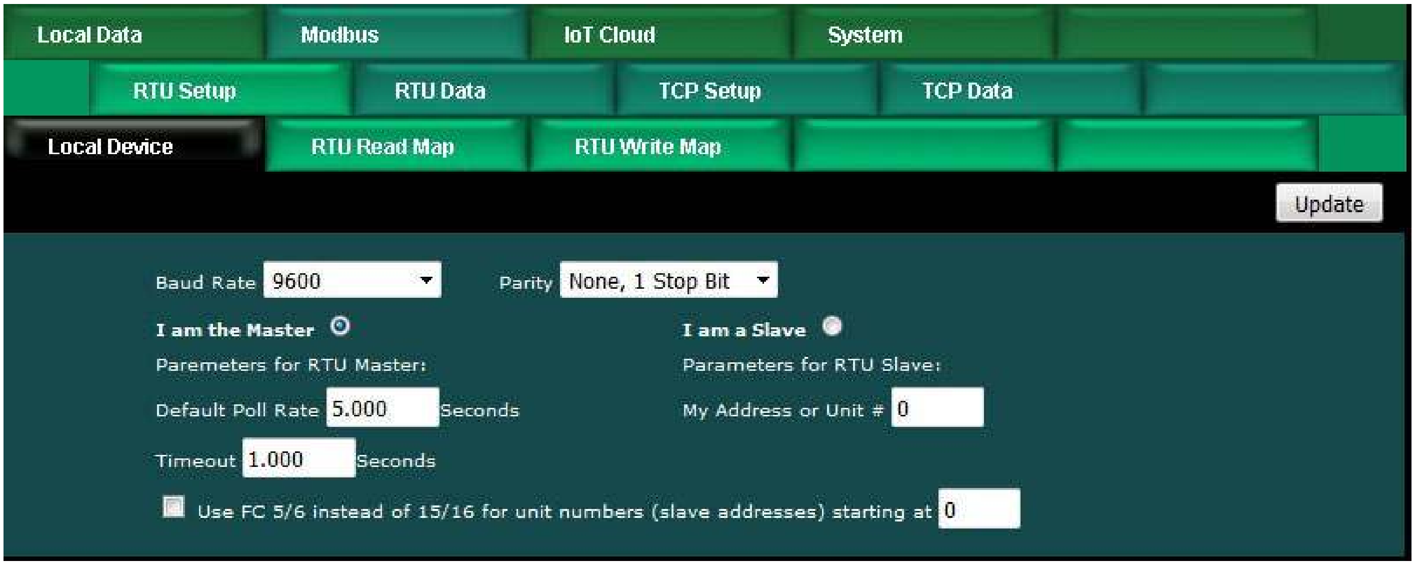
As noted in the previous example, the Babel Buster IoT Gateway provides support for character strings as a series of Modbus registers that are treated as a single piece of data just like 32-bit integers are a pair of registers treated as one. The most powerful aspect of this feature is the ease of creating and manipulating character strings in Script Basic. Here is another example.



File: /FS/FLASH0/display1.sb

```
setreg 21, "Hello from Basic"
setreg 41, "This is a demo"
setreg 61, "of writing display"
setreg 81, "from program."
```

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 Modus RTU port for the baud rate that matches the display. The Babel Buster will be the master since the display is an RTU slave.



Baud Rate 9600 Parity None, 1 Stop Bit

I am the Master ● I am a Slave ○

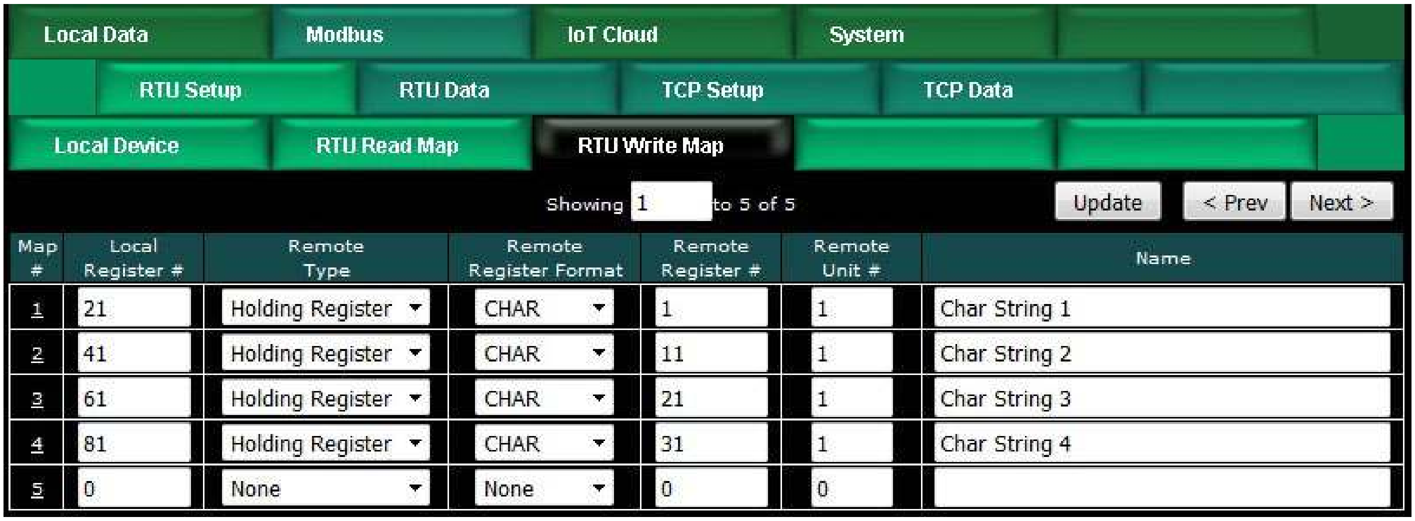
Paremeters for RTU Master: Parameters for RTU Slave:

Default Poll Rate 5.000 Seconds My Address or Unit # 0

Timeout 1.000 Seconds

□ 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.



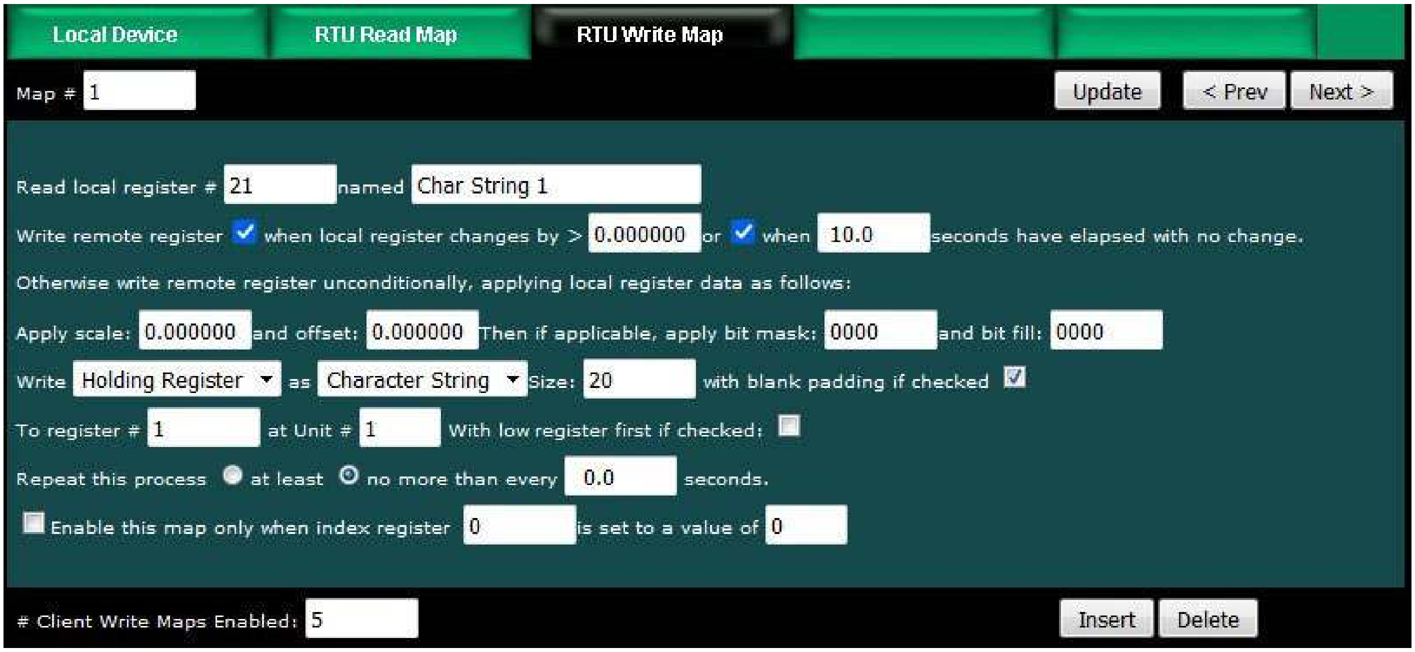
Showing 1 to 5 of 5

Map #	Local Register #	Remote Type	Remote Register Format	Remote Register #	Remote Unit #	Name
1	21	Holding Register	CHAR	1	1	Char String 1
2	41	Holding Register	CHAR	11	1	Char String 2
3	61	Holding Register	CHAR	21	1	Char String 3
4	81	Holding Register	CHAR	31	1	Char String 4
5	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 Babel Buster 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.



Map # 1

Read local register # 21 named Char String 1

Write remote register ☑ when local register changes by > 0.000000 or ☑ when 10.0 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 Character String Size: 20 with blank padding if checked ☑

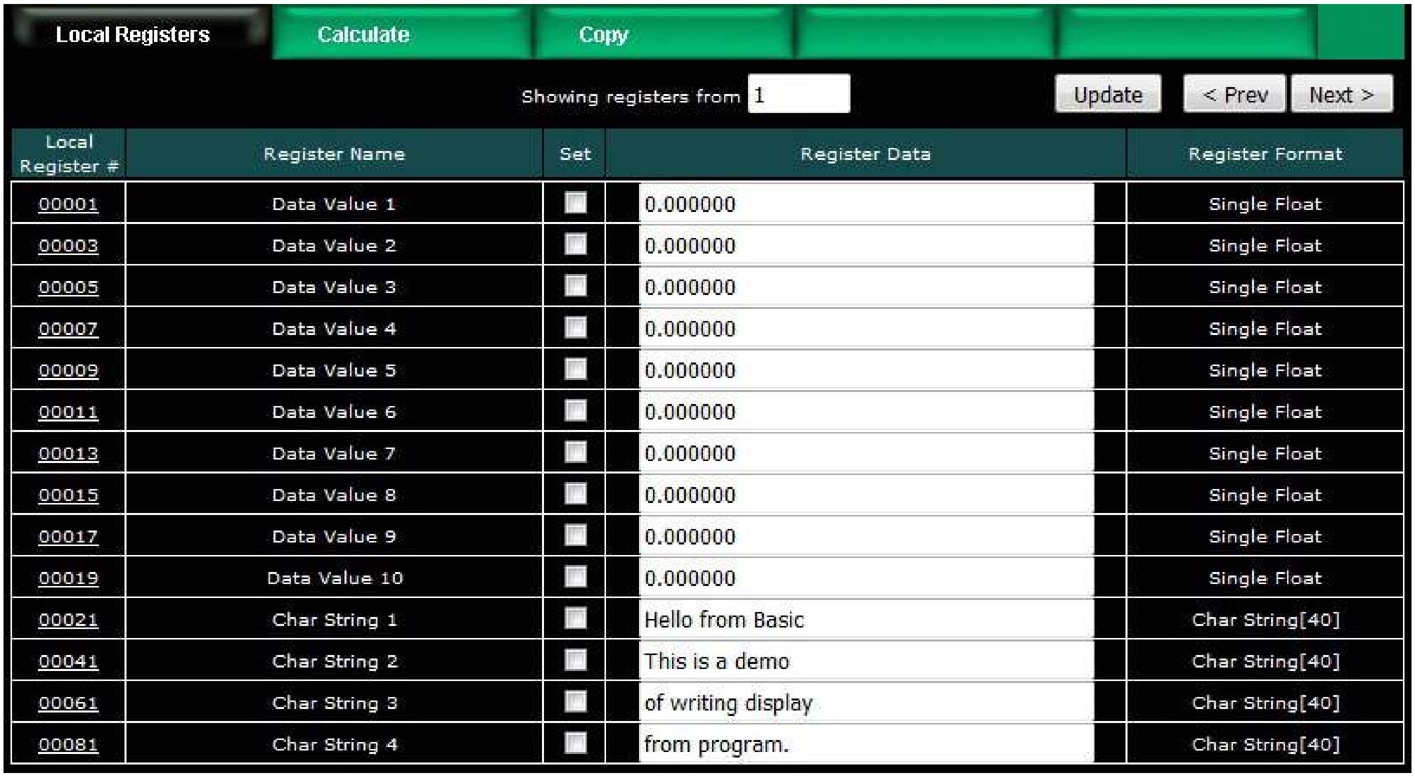
To register # 1 at Unit # 1 With low register first if checked: □

Repeat this process ○ at least ● no more than every 0.0 seconds.

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

Client Write Maps Enabled: 5

If we click Start on the Programming Build/Run page, the character registers will be set as illustrated below.



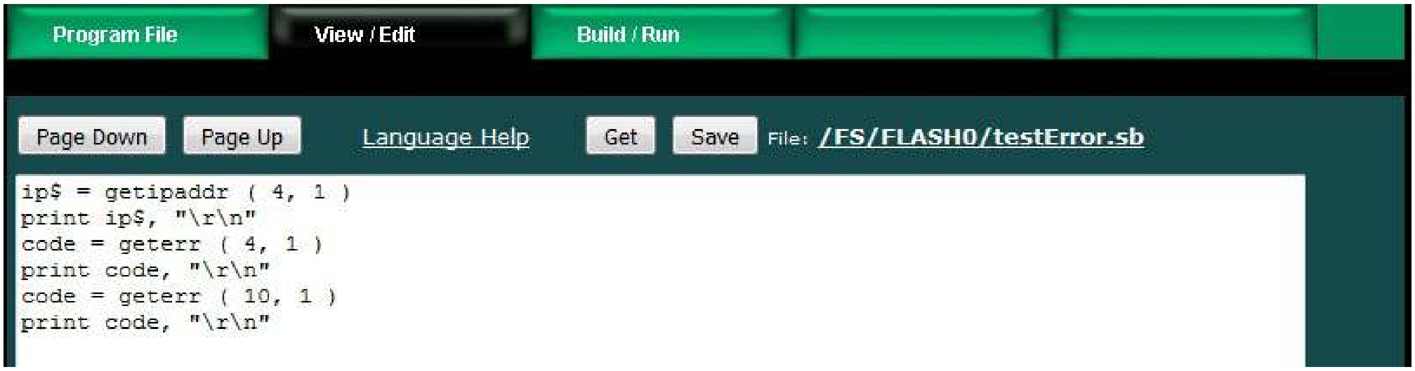
Showing registers from 1

Local Register #	Register Name	Set	Register Data	Register Format
00001	Data Value 1	□	0.000000	Single Float
00003	Data Value 2	□	0.000000	Single Float
00005	Data Value 3	□	0.000000	Single Float
00007	Data Value 4	□	0.000000	Single Float
00009	Data Value 5	□	0.000000	Single Float
00011	Data Value 6	□	0.000000	Single Float
00013	Data Value 7	□	0.000000	Single Float
00015	Data Value 8	□	0.000000	Single Float
00017	Data Value 9	□	0.000000	Single Float
00019	Data Value 10	□	0.000000	Single Float
00021	Char String 1	□	Hello from Basic	Char String[40]
00041	Char String 2	□	This is a demo	Char String[40]
00061	Char String 3	□	of writing display	Char String[40]
00081	Char String 4	□	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.

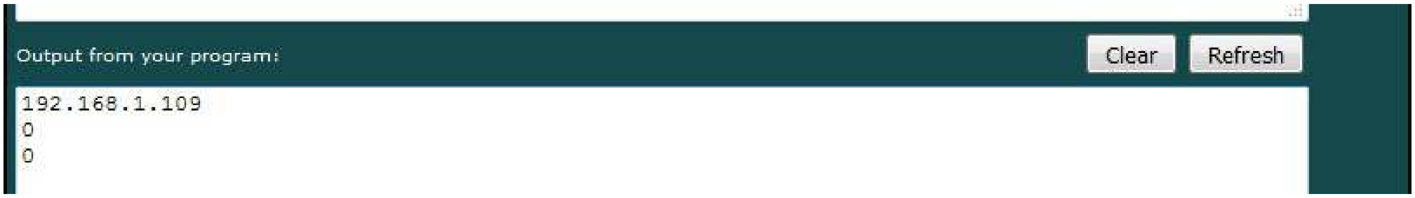
19.7 Example: Device and Rule Diagnostics

Here is a short example illustrating the special diagnostic functions available in Script Basic. In this example, we are checking on our Babel Buster'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.



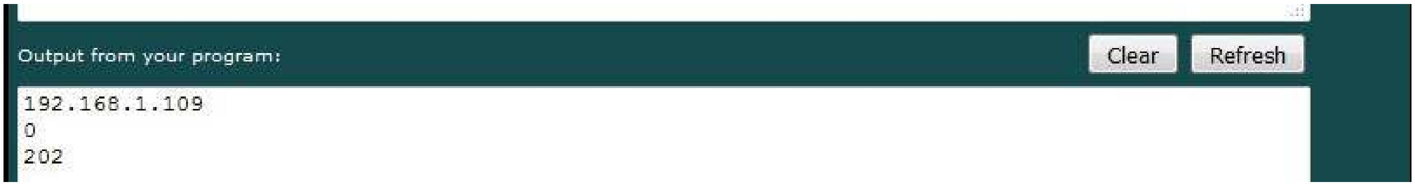
```
ip$ = getipaddr ( 4, 1 )
print ip$, "\r\n"
code = geterr ( 4, 1 )
print code, "\r\n"
code = geterr ( 10, 1 )
print code, "\r\n"
```

Normal results will be to display the IP address and two error codes that will be zero if there are no problems.



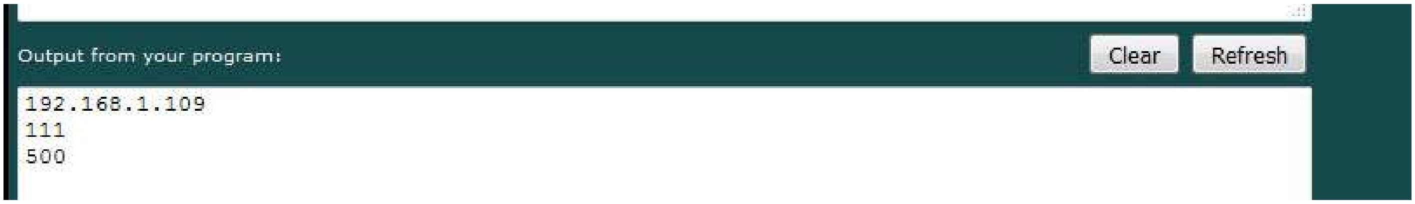
```
192.168.1.109
0
0
```

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.



```
192.168.1.109
0
202
```

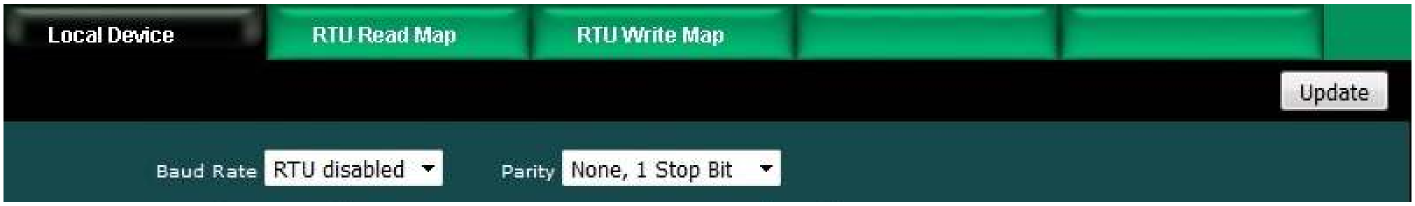
This example was created using ModSim as the Modbus TCP slave. By terminating the ModSim program, we no longer have any Modbus TCP device accessible. This results in a timeout indicated by the read map (500) . The device status is indicating 111, which means the IP address is reachable, but Modbus TCP cannot be found there - and this is the expected error in this deliberately flawed example.



```
192.168.1.109
111
500
```

19.8 Example: Connecting Custom Serial Device to AWS

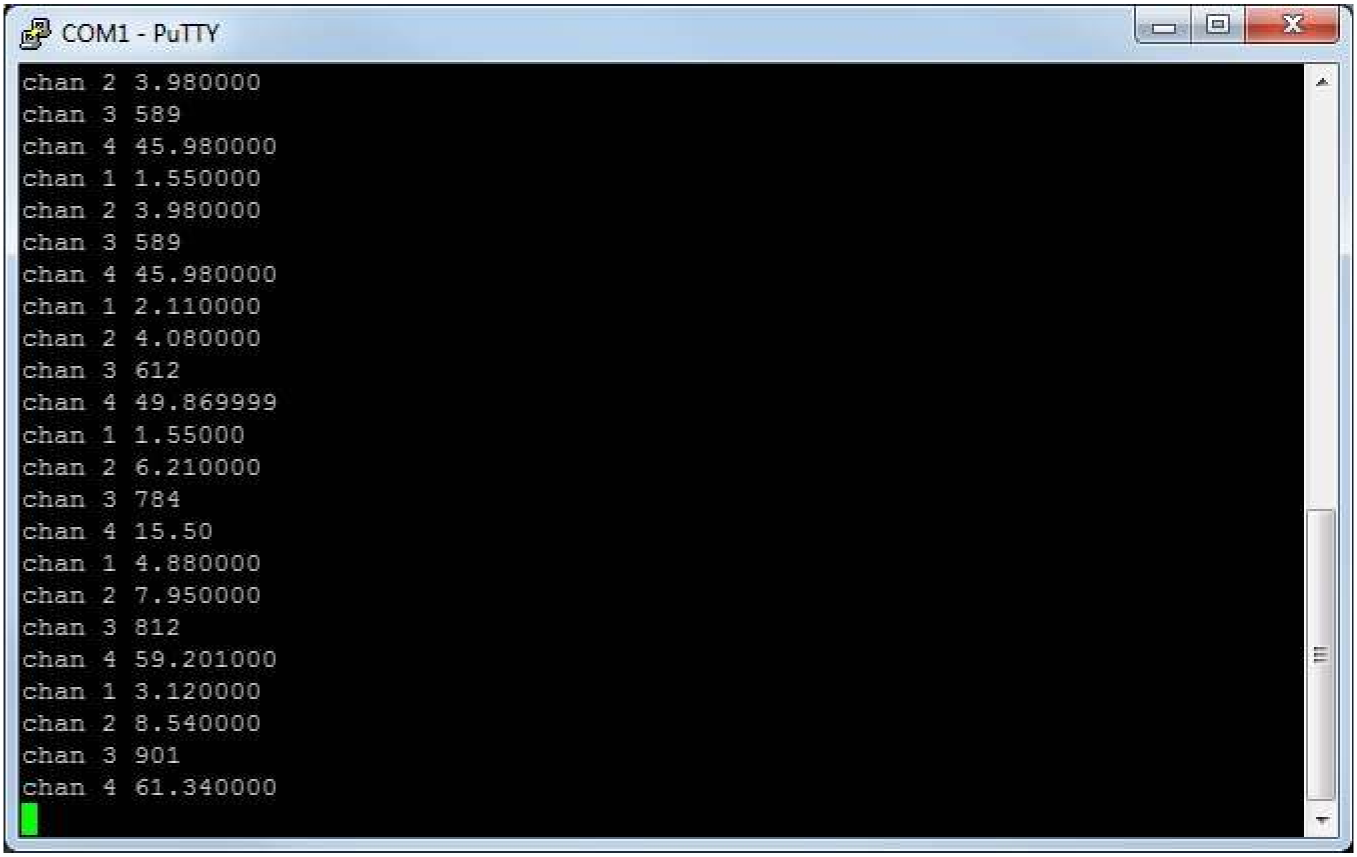
A very useful feature available with the Babel Buster is the ability to disable Modbus RTU and use the serial port for a proprietary serial protocol instead. When Modbus RTU is disabled, the baud rate of the port is set by Basic. The proprietary protocol may be one where you need to send commands to 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.



Baud Rate RTU disabled

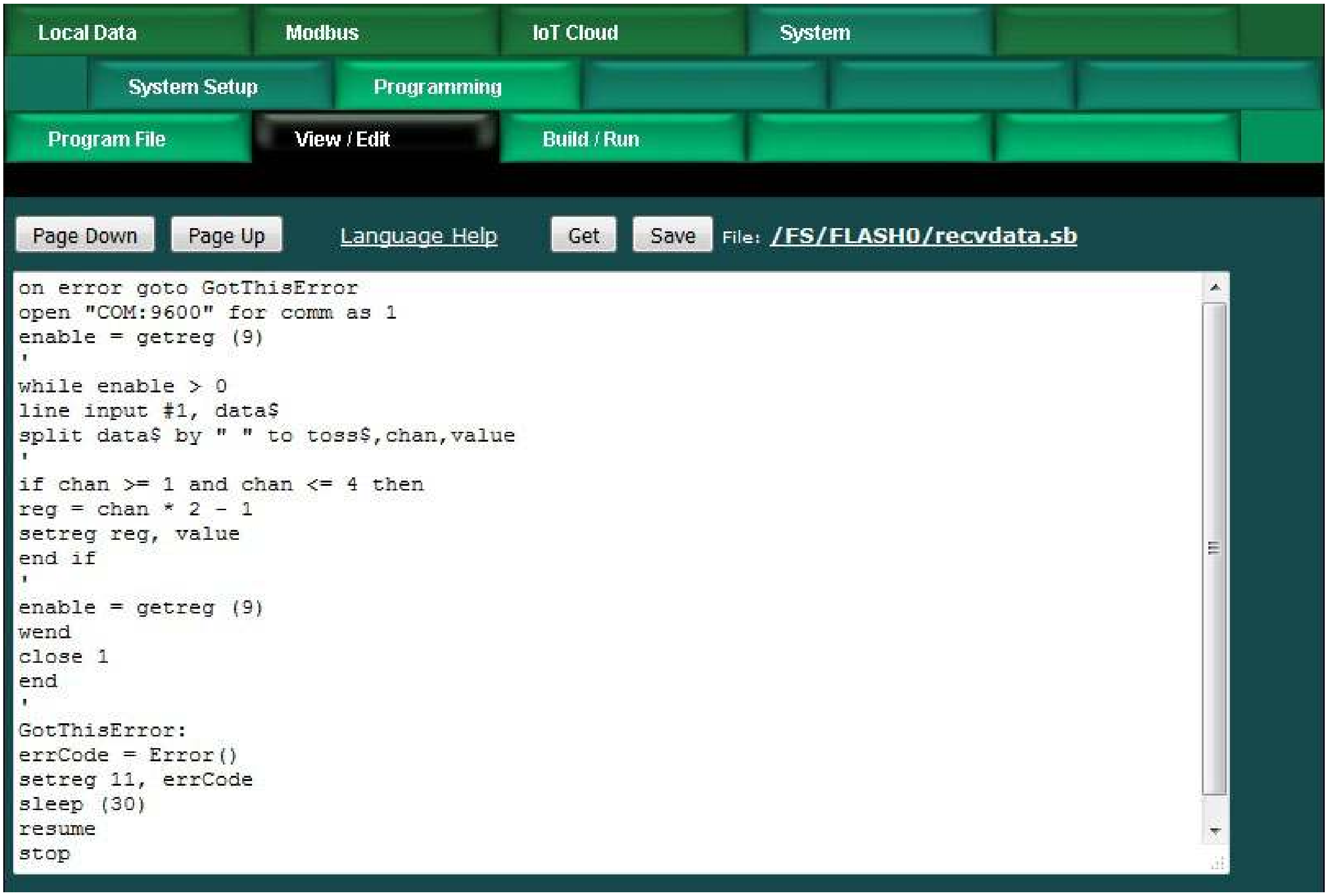
Parity None, 1 Stop Bit

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.



```
chan 2 3.980000
chan 3 589
chan 4 45.980000
chan 1 1.550000
chan 2 3.980000
chan 3 589
chan 4 45.980000
chan 1 2.110000
chan 2 4.080000
chan 3 612
chan 4 49.869999
chan 1 1.55000
chan 2 6.210000
chan 3 784
chan 4 15.50
chan 1 4.880000
chan 2 7.950000
chan 3 812
chan 4 59.201000
chan 1 3.120000
chan 2 8.540000
chan 3 901
chan 4 61.340000
```

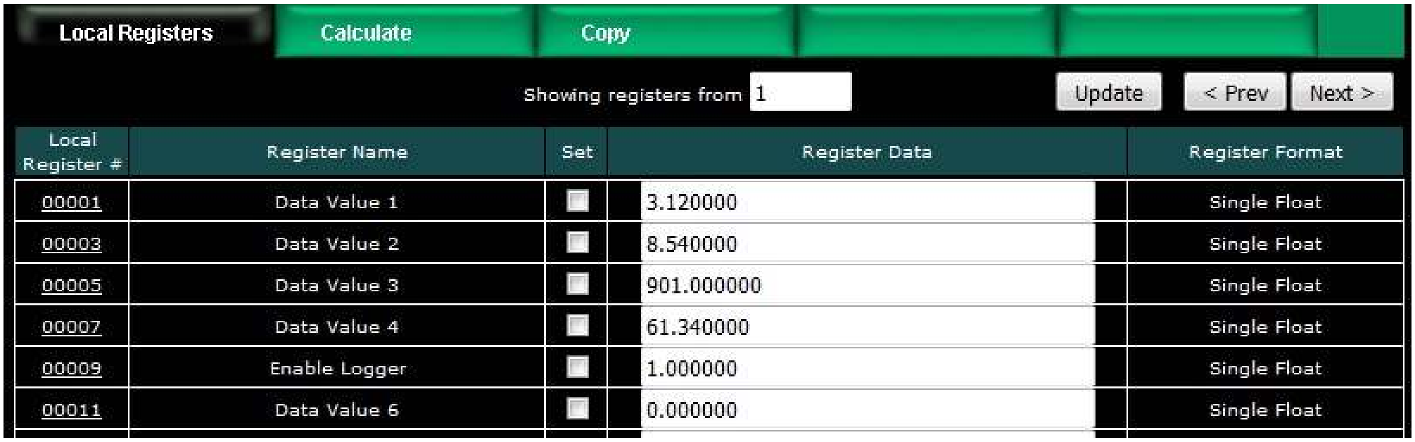
The program to capture data from this device and store the results in local registers is illustrated below. The key line to notice here is the "split" where the data line is effectively parsed. The "toss$" is going to end up holding the string "chan" which we don't care about. The numbers representing 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.



File: /FS/FLASH0/recvdata.sb

```
on error goto GotThisError
open "COM:9600" for comm as 1
enable = getreg (9)
'
while enable > 0
line input #1, data$
split data$ by " " to toss$,chan,value
'
if chan >= 1 and chan <= 4 then
reg = chan * 2 - 1
setreg reg, value
end if
'
enable = getreg (9)
wend
close 1
end
'
GotThisError:
errCode = Error()
setreg 11, errCode
sleep (30)
resume
stop
```

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.



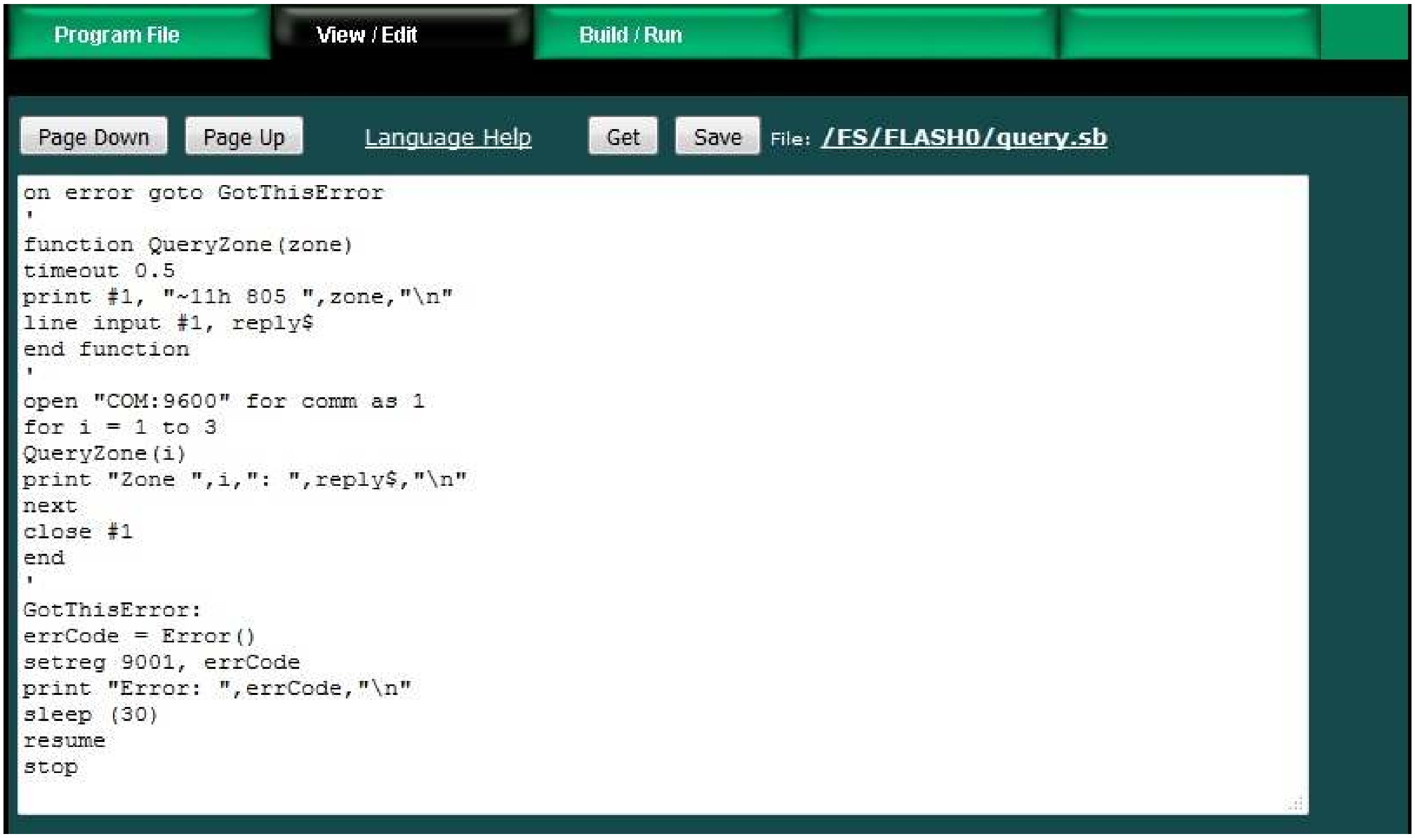
Showing registers from 1

Local Register #	Register Name	Set	Register Data	Register Format
00001	Data Value 1	□	3.120000	Single Float
00003	Data Value 2	□	8.540000	Single Float
00005	Data Value 3	□	901.000000	Single Float
00007	Data Value 4	□	61.340000	Single Float
00009	Enable Logger	□	1.000000	Single Float
00011	Data Value 6	□	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.

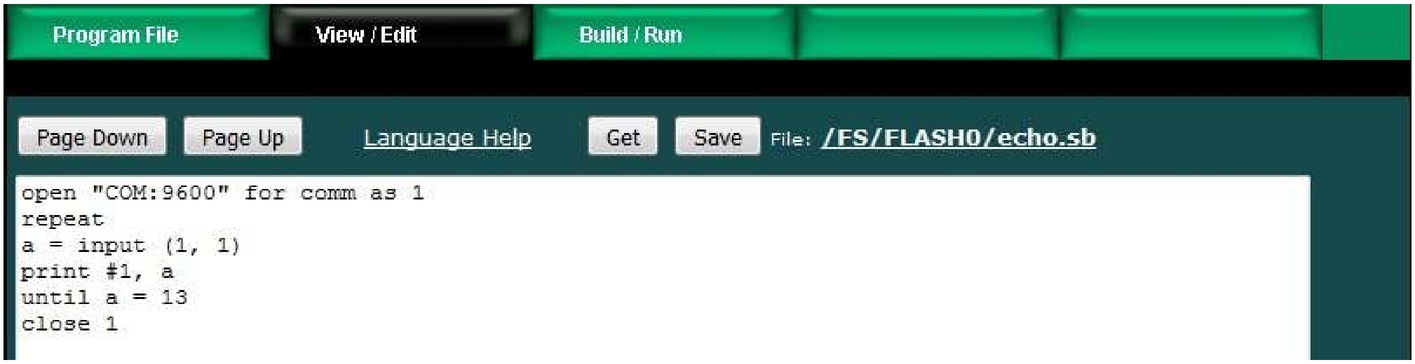
The above example is always just receiving data that is sent automatically. If you needed to query a device in order to receive data, you can do that from Script Basic. The following example shows querying one particular type of device. The string printed in the QueryZone function is unique to this device - 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).



```
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.



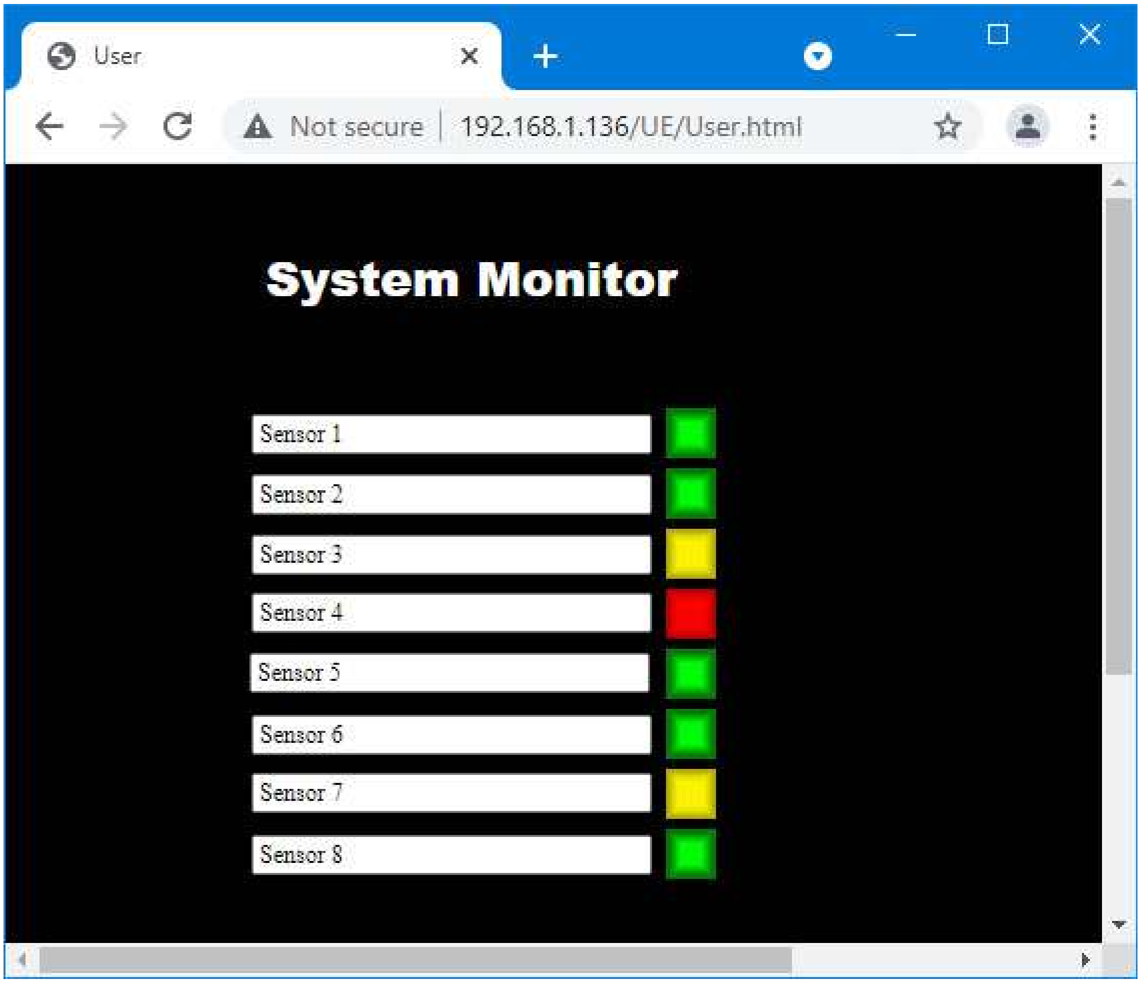
```
open "COM:9600" for comm as 1
repeat
a = input (1, 1)
print #1, a
until a = 13
close 1
```

Babel Buster IoT
MODEL MQ-61
MODEL BB3-6101-MQ
IOT GATEWAY

CONTROL SOLUTIONS MINNESOTA

20. 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 Babel Buster 3 gateways. Except for changing the link "../index.html" to "/html/index.html" where applicable, the various demos found in the knowledgebase should work in a Babel Buster 3 (including MQ-61).



System Monitor

Sensor 1 🟩
Sensor 2 🟩
Sensor 3 🟨
Sensor 4 🟥
Sensor 5 🟩
Sensor 6 🟩
Sensor 7 🟨
Sensor 8 🟩

The "naked pages" referenced in the 2008 version of the CGI overview are not implemented in Babel Buster 3. They were an attempt at allowing some level of "private labeling" the gateway. If private labeling is desired, Control Solutions does offer that option which will result in fully rebranding the entire preprogrammed web UI.

20.1 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. 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 recently tested FileZilla v3.54.1 and it seemed to work ok.

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/.

20.2 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:

```
    <img src="specialbutton.gif" onclick="sendMeAway();" alt=""
height="25" width="133" border="0">

    <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:

```
    <img src="/UE/isotech.gif" alt="" height="448" width="804"
border="0">
```

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 not *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:

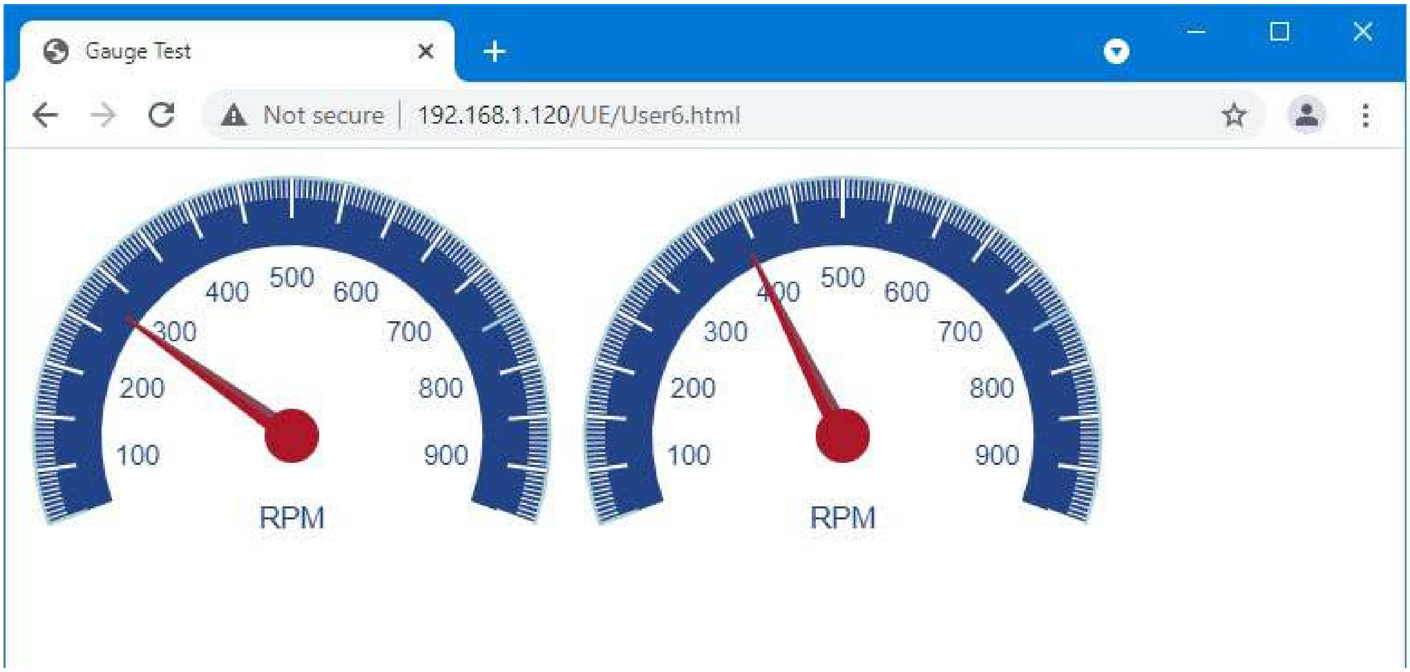
```
    <input type="submit" name="submitSave" value="Save">
    <input type="submit" name="submitRestart" value="Restart">
```

20.3 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 animageGauges() 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 the MQ-61/BB3-6101-MQ. The iframe example will work on any Babel Buster 3 family device supporting User HTML. The REST API option is only available on certain models. The gauge animation could be restructured to take advantage of the REST API if you know your way around JavaScript.

Babel Buster IoT
MODEL MQ-61
MODEL BB3-6101-MQ
IOT GATEWAY

CONTROL SOLUTIONS MINNESOTA

21. REST API

You can use the IoT Gateway 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.

21.1 GET Data from Device

You can use an HTTP GET request to query the IoT gateway 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".

21.2 POST Data to Device

To write data to the IoT gateway (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.

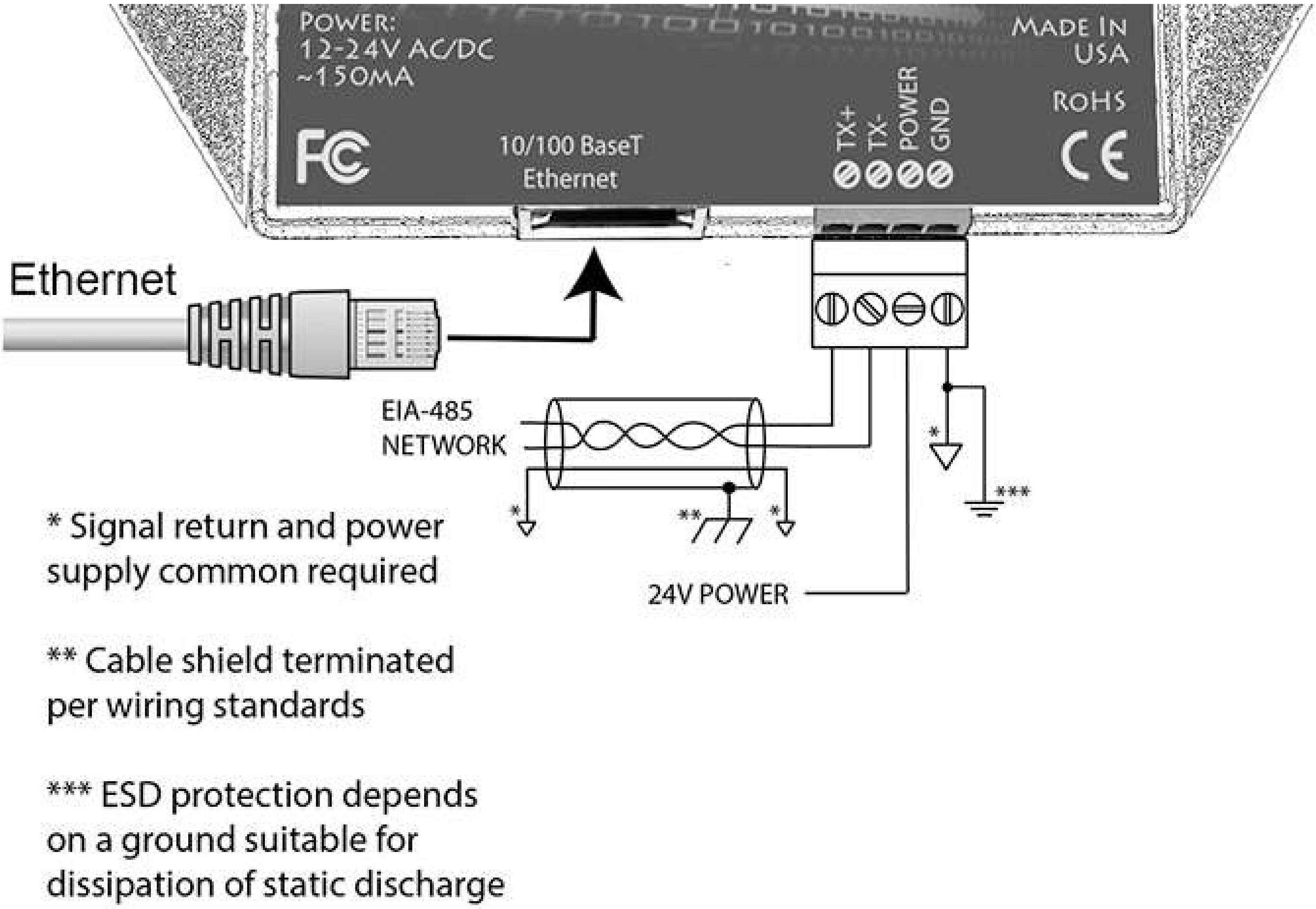
Babel Buster IoT
MODEL MQ-61
MODEL BB3-6101-MQ
IOT GATEWAY

CONTROL SOLUTIONS MINNESOTA

Appendix A Hardware Details

A.1 Wiring

Wiring for the Babel Buster MQ-61 is illustrated below.

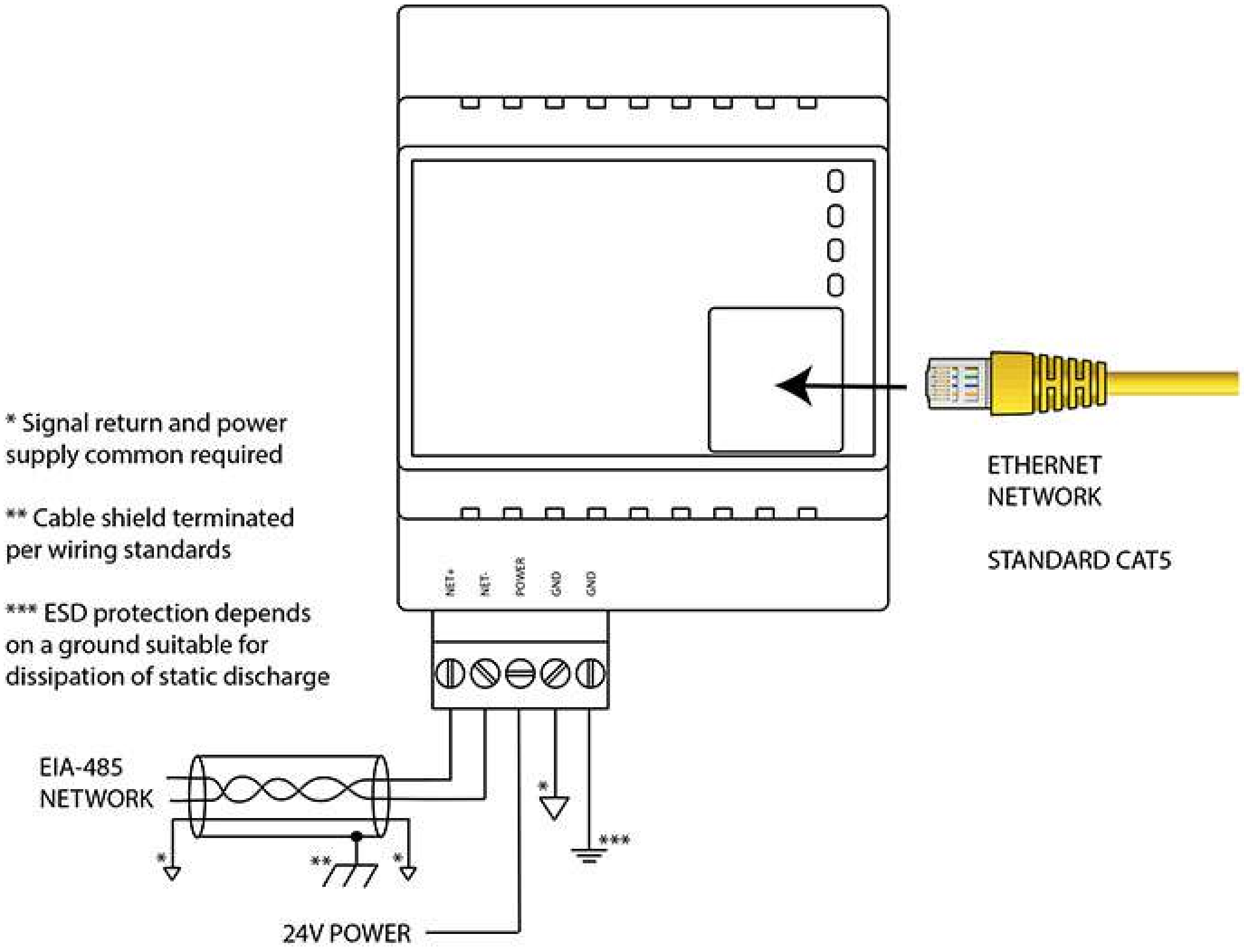


* Signal return and power supply common required

** Cable shield terminated per wiring standards

*** ESD protection depends on a ground suitable for dissipation of static discharge

Wiring for the Babel Buster BB3-6101-MQ is illustrated below:


* Signal return and power supply common required
** Cable shield terminated per wiring standards
*** ESD protection depends on a ground suitable for dissipation of static discharge
ETHERNET NETWORK
STANDARD CAT5
NET+
NET-
POWER
GND
GND
EIA-485 NETWORK
*
**
*
*
***
24V POWER

Wire the gateway as illustrated. Follow all conventional standards for wiring of EIA-485 networks when connecting the Modbus RTU EIA-485 (RS485) network. This includes use and termination of shield, termination of the network, and grounding.

IMPORTANT: Although EIA-485 (RS485) is thought of as a 2-wire network, you MUST include a third conductor connected to GND or common at each device so that all devices are operating at close to the same ground potential. Proper grounding of equipment should ensure proper operation without the third conductor; however, proper grounding often cannot be relied upon. If large common mode voltages are present, you may even need to insert optically isolated repeaters between EIA-485 devices.

Use standard CAT5 cables for Ethernet connections. Use control wire as applicable for local electrical codes for connecting the 24V (AC or DC) power supply.

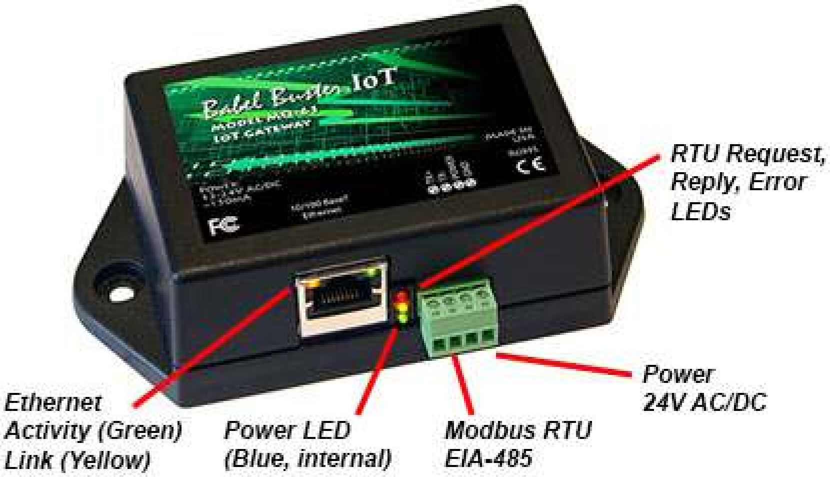
Note that in addition to connecting power supply common to a GND terminal, you must also connect a GND terminal to earth ground in order to ensure proper ESD protection.

BB3-6101-MQ-232: The standard BB3-6101 Modbus RTU port uses RS-485. The RS-232 version replaces the RS-485 transceiver with an RS-232 transceiver. The

NET+/NET- terminals are replaced by TXD and RXD on the -232 version. TXD is data out from the BB3-6101-232, and RXD is data in to the gateway. Hardware handshake is not supported.

A.2 Front Panel LED Indicators

MQ-61 Power-up LED behavior: Following a server boot-up delay, all LEDs on front panel will turn on for half a second. Then they will proceed to indicate as normally defined for the indicators.


RTU Request, Reply, Error LEDs
Ethernet Activity (Green) Link (Yellow)
Power LED (Blue, internal)
Modbus RTU EIA-485
Power 24V AC/DC

Babel Buster MQ-61 request/reply LEDs reflect RTU traffic while the Ethernet activity LED will indicate TCP traffic. To see TCP errors, one needs to look at the Errors page in the web UI.

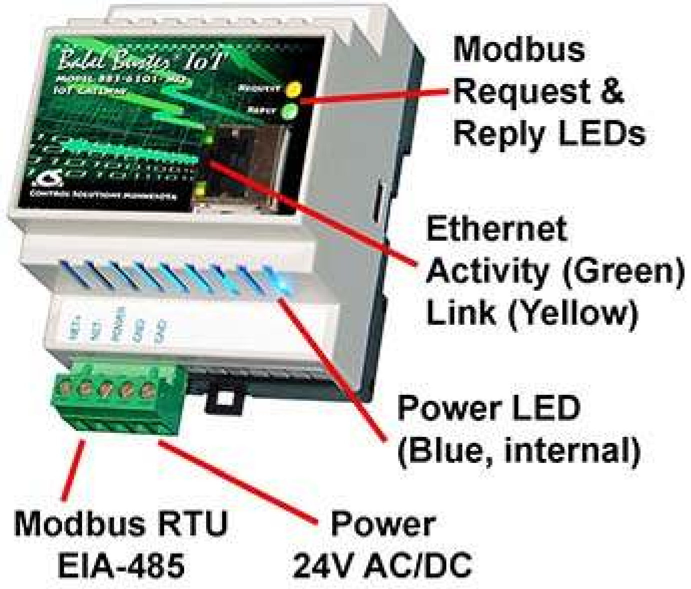
Babel Buster MQ-61 LEDs indicate as follows:

Error (red)	Operating as Modbus Master, flashes 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 red if an exception code is sent (meaning the received request resulted in an error).
Request (yellow)	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 (green)	Operating as Modbus Master, flashes green each time a good response is received. Operating as Modbus Slave, flashes green each time a good response is sent.
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.

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

BB3-6101-MQ 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.


Modbus Request & Reply LEDs
Ethernet Activity (Green) Link (Yellow)
Power LED (Blue, internal)
Modbus RTU EIA-485
Power 24V AC/DC

Babel Buster BB3-6101 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.

Babel Buster BB3-6101 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.

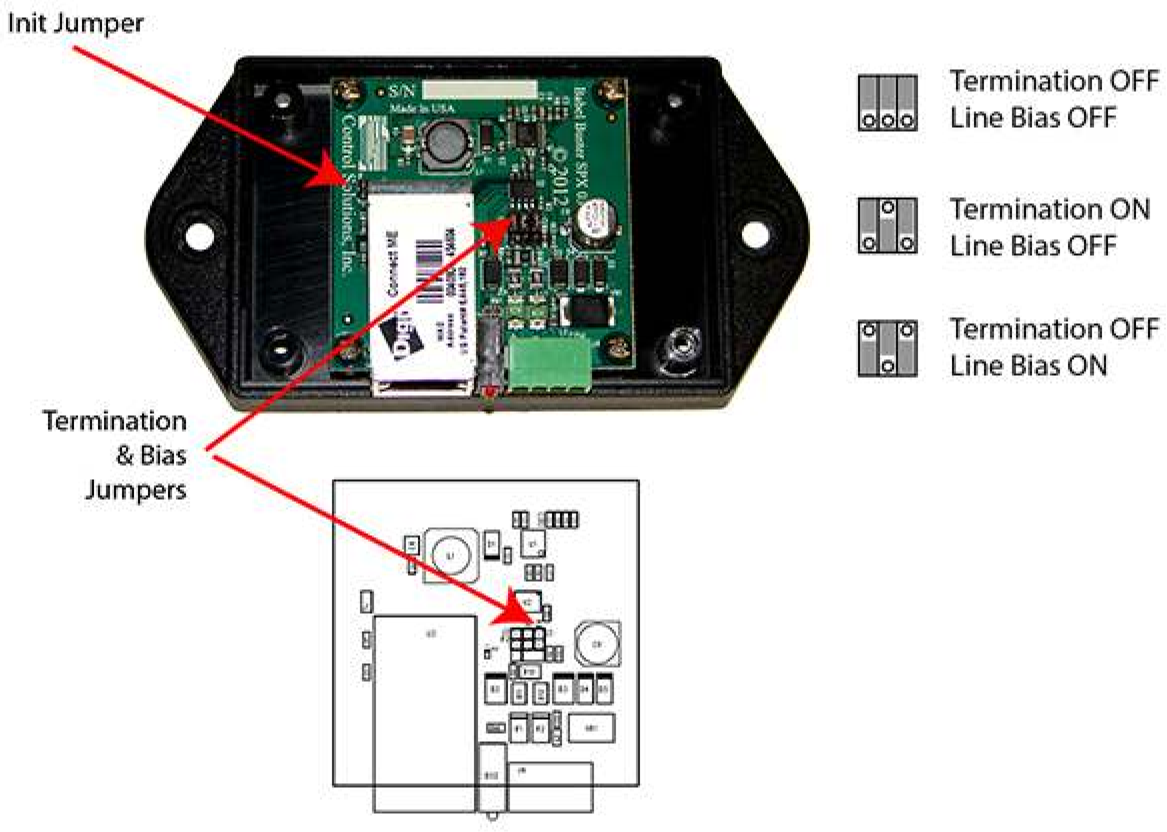
A.3 RS-485 Line Termination & Bias

Enable line termination only when this device is placed at the end of the network. Termination should only be enabled at two points on the network, and these two points must be specifically the end points.

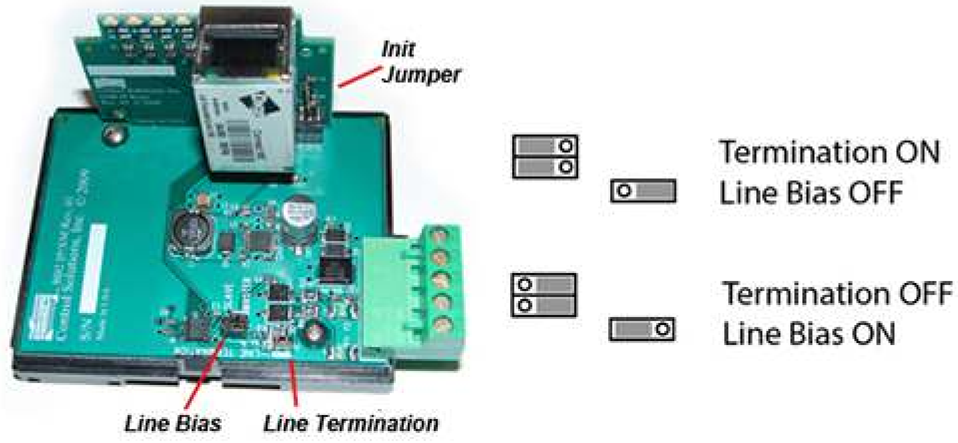
Enable line bias when needed. Line bias should only be enabled at one point on the network, and does not have to be the end point. Line bias holds the line in a known neutral state when no devices are transmitting. Without bias, the transition from offline to online by a transmitter can look like a false start bit and cause loss of communication.

The line conditioning options are enabled when the respective shunt is moved to the position indicated by the diagram below.

Jumper locations for Babel Buster MQ-61:



Jumper locations for the Babel Buster BB3-6101-MQ:



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 gateway'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 gateway 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

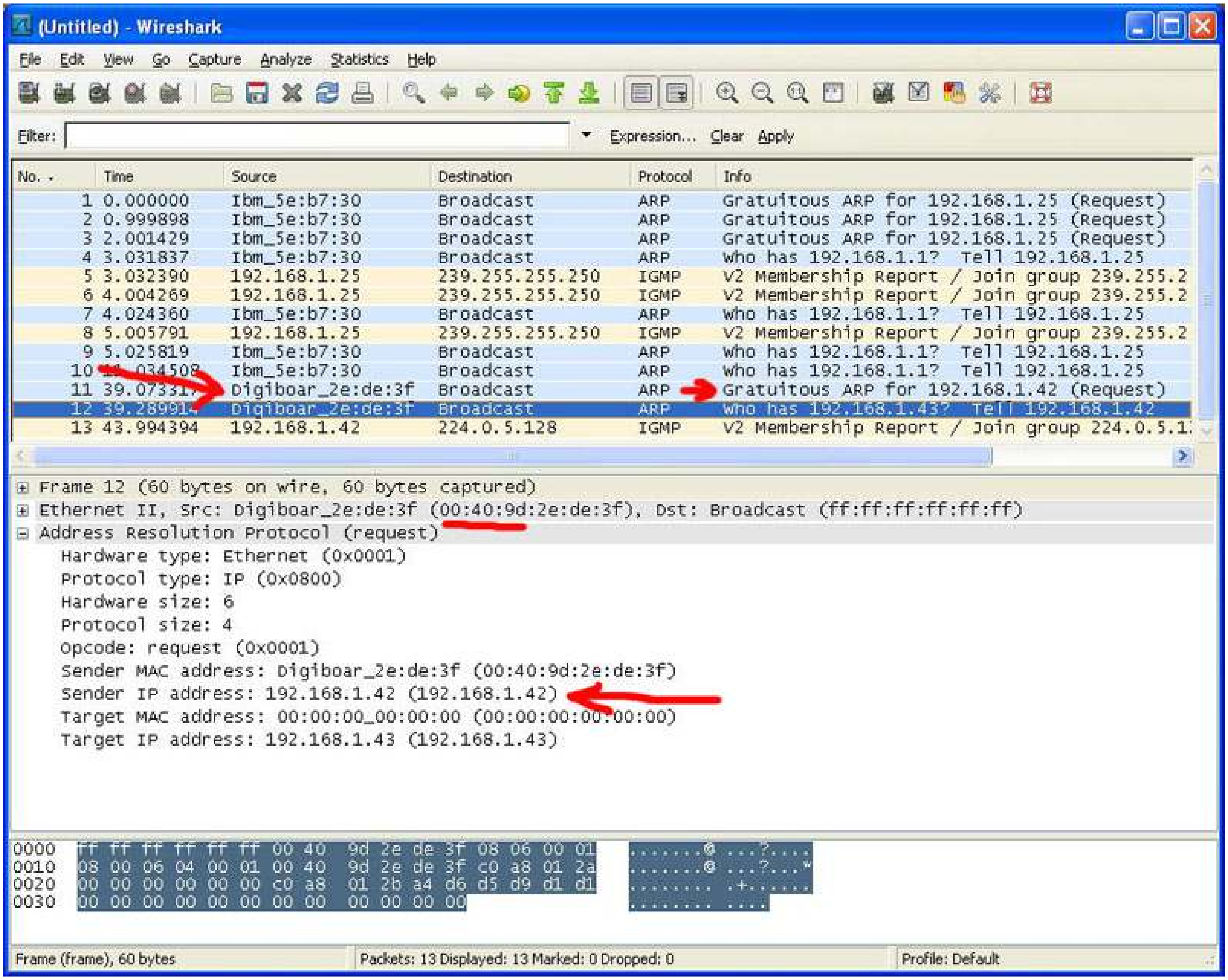
You can use Wireshark to discover a lost IP address if the gateway is still functional. Connect the gateway directly to your PC running Wireshark using a cross-over cable (or standard CAT5 cable if your PC supports auto-MDX). With Wireshark running, power up the gateway.

Upon power up, MQ-61/BB3-6101-MQ 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 gateway 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 gateway 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 gateway 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 [illegible].034508    Ibm_5e:b7:30        Broadcast          ARP       Who has 192.168.1.1?  Tell 192.168.1.25
     11 39.07331[illegible]    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        224.0.5.128        IGMP      V2 Membership Report / Join group 224.0.5.1:

⊞ 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 MQ-61/BB3-6101-MQ serves two purposes, and what it does depends on whether you apply the jumper before or after the gateway boots up.

Hard Configuration Reset:

Installing the jumper after bootup causes the gateway 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 gateway 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 repeate the allocation setup as described in Section 3.4.

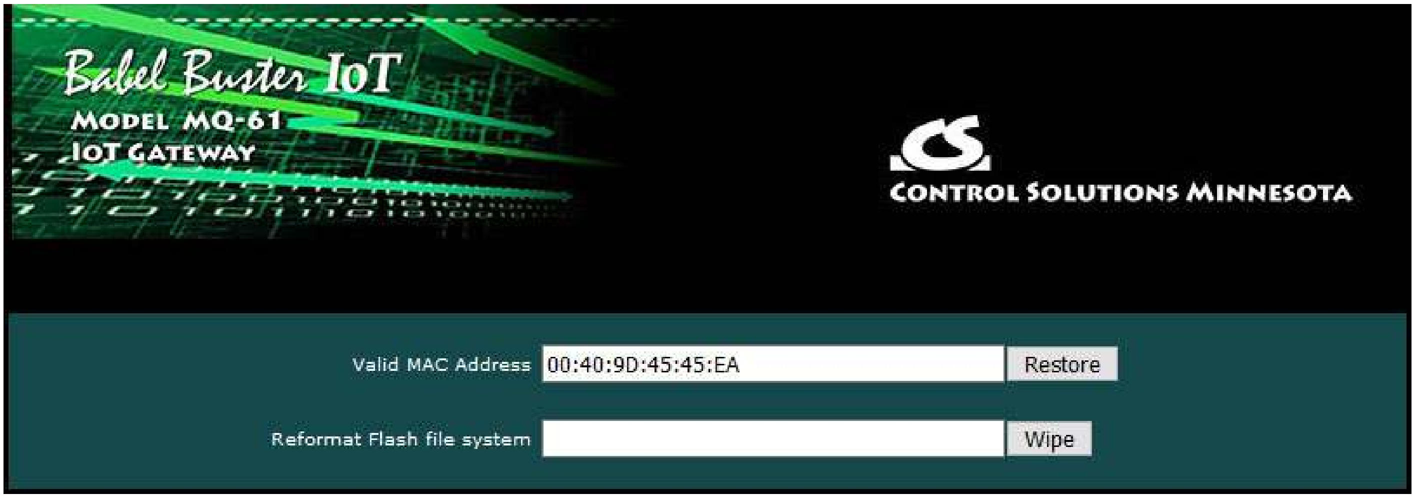
Firmware Update Recovery:

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 gateway'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 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.



Valid MAC Address 00:40:9D:45:45:EA

Reformat Flash file system

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 gotton 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 procedes 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 gateway 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.

Appendix B Modbus CSV Import Files

B.1 Modbus RTU Master Read/Write Maps

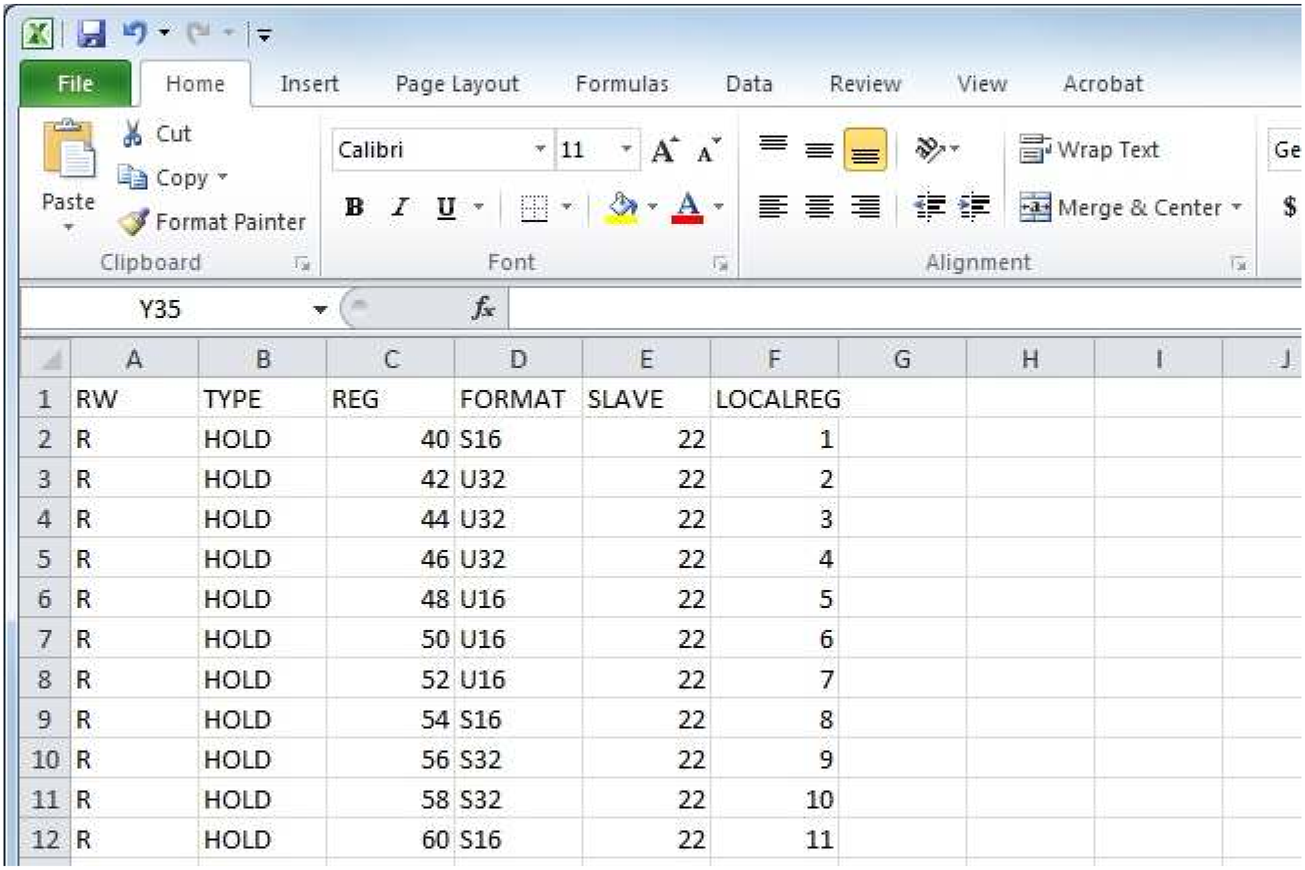
The CSV file for configuring Modbus TCP client read and write maps should contain a single header line with the labels indicated below, and content as applicable.

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 Babel Buster gateways 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 the retained bits will be right justified in the result. 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.
DEFAULT	-	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. 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.
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.

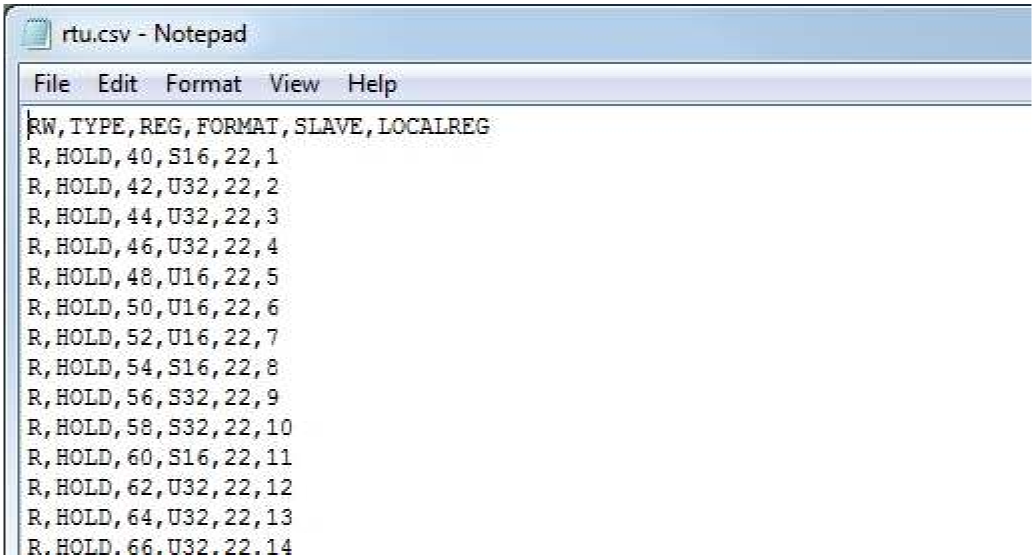
The minimum required header line for Modbus RTU must include RW, LOCALREG, TYPE, REG, FORMAT, and SLAVE. All other columns are optional.

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:



```
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	-	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. 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.
DEFAULT	-	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. 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.

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

Babel Buster IoT
MODEL MQ-61
MODEL BB3-6101-MQ
IOT GATEWAY

CONTROL SOLUTIONS MINNESOTA

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 Babel Buster IoT Gateway is configured as Modbus master, then the Error Counts page will list counts by slave address. If the Babel Buster is configured as Modbus slave, then errors show up on the first line (Unit # 1) regardless of what address the Babel Buster is configured to be.

The Errors: Read Maps and Errors: Write Maps pages will tell you exactly which maps are getting errors when the Babel Buster 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-458 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 Babel Buster 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 Babel Buster 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 Babel Buster 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 Babel Buster 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 IoT Gateway 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 IoT Gateway, 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 IoT Gateway 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 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.
4	Slave Device Failure	An unrecoverable error occurred (for BB2-3060 means corrupt packet was received).

6	Slave Device Busy	The slave is engaged in processing a long-duration command. The master should try again later.
10 (hex 0A)	Gateway Path Unavailable	Gateway could not establish communication with target device. In the case of BB2-3060, will most often mean there is no IP address configured.
11 (hex 0B)	Gateway Target Device Failed to Respond	Specialized use in conjunction with gateways, indicates no response was received from the target device. In the case of BB2-3060, means there was a TCP link failure, unable to connect to TCP target device.
17 (hex 11)	Gateway Target Device Failed to Respond	No response from slave, request timed out.

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 Babel Buster gateways, 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 Babel Buster gateways 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 Babel Buster gateways 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 Babel Buster gateways 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 gateways, 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). Babel Buster gateways 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. Babel Buster gateways 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 Babel Buster 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 back 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 Babel Buster gateway 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 gateways on the same Modbus network?

You can not have more than one Master on a Modbus RTU (RS-485) network. Therefore, if the gateway is to be configured as the Master, you can only have 1 gateway. You cannot use multiple gateways to read more points from the same Modbus slave device.

Multiple gateways 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?

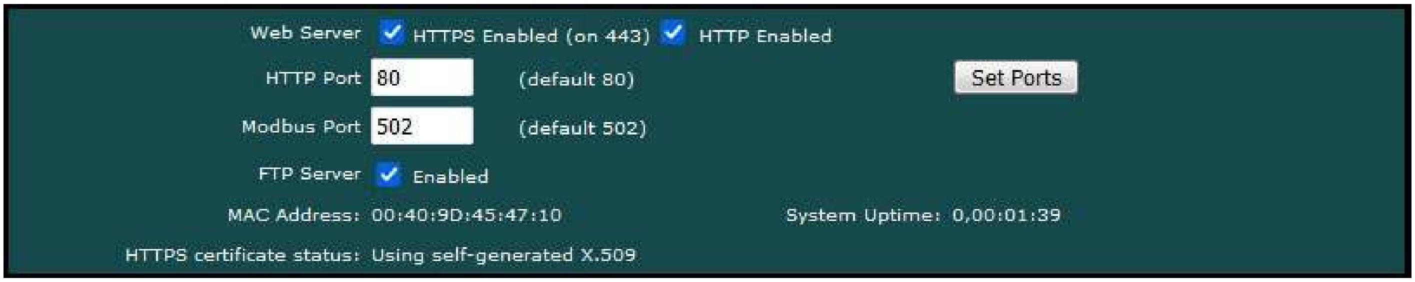
Logically you can address over 250 devices; however, the RS-485 transceivers are not capable of physically driving that many devices. Modbus protocol states that the limit is 32 devices, and most RS-485 transceivers will agree with this. Only if all devices on the network have low load transceivers can 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 Babel Buster IoT Gateway.

D.1 X.509 Auto-Certificate Generation

The Babel Buster IoT Gateway 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.



Web Server ☑ HTTPS Enabled (on 443) ☑ HTTP Enabled

HTTP Port 80 (default 80)

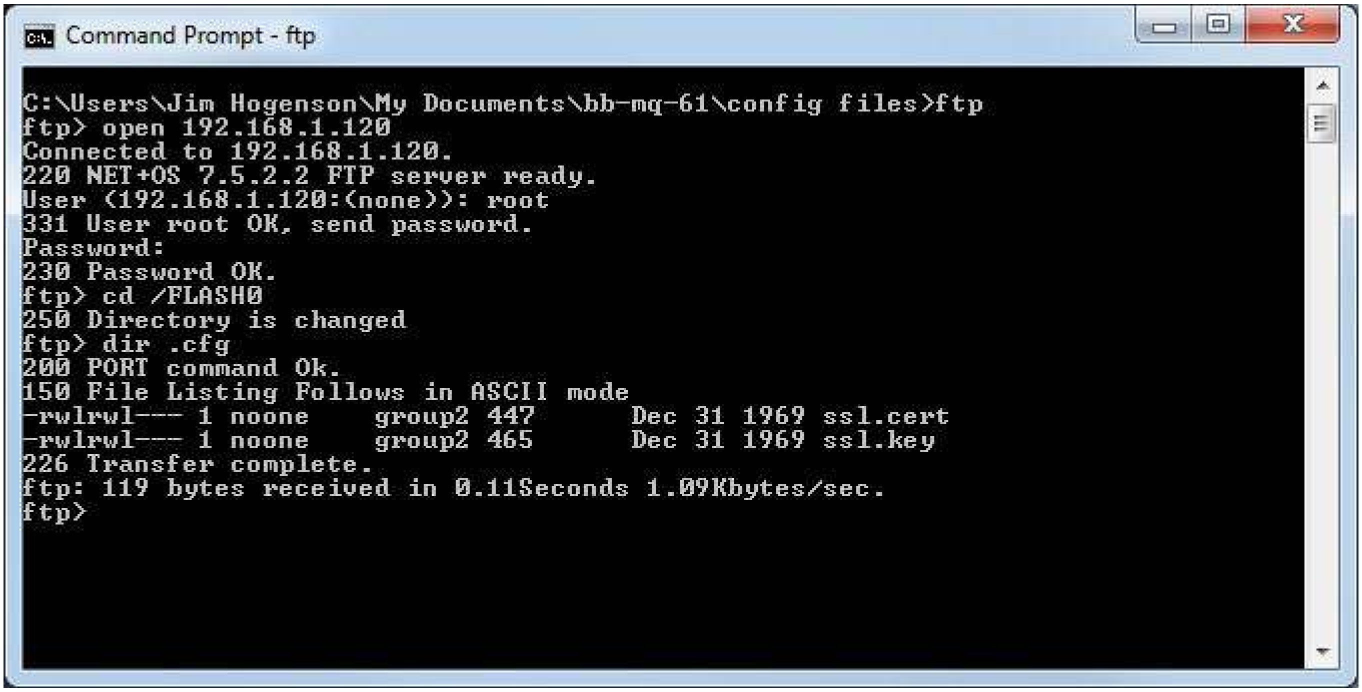
Modbus Port 502 (default 502)

FTP Server ☑ Enabled

MAC Address: 00:40:9D:45:47:10 System Uptime: 0,00:01:39

HTTPS certificate status: Using self-generated X.509

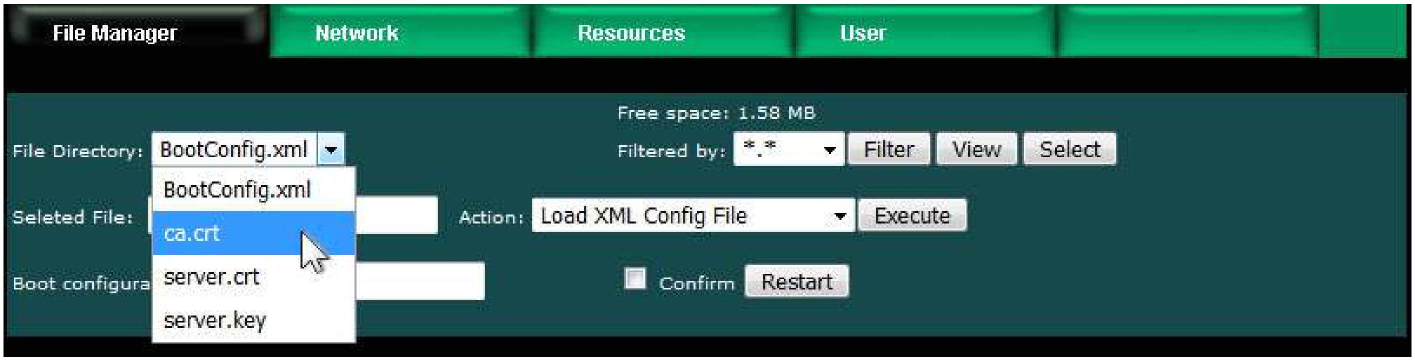
If there is a need to delete the self-generated certificates, you can do so by logging in via FTP. Change directory to /FLASH0, then to .cfg. The two certificate files that were self-generated are ssl.cert and ssl.key.



```
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
-rwlrwl--- 1 noone    group2 447      Dec 31 1969 ssl.cert
-rwlrwl--- 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.



Free space: 1.58 MB

File Directory: BootConfig.xml

Filtered by: *.*

Seleted File:

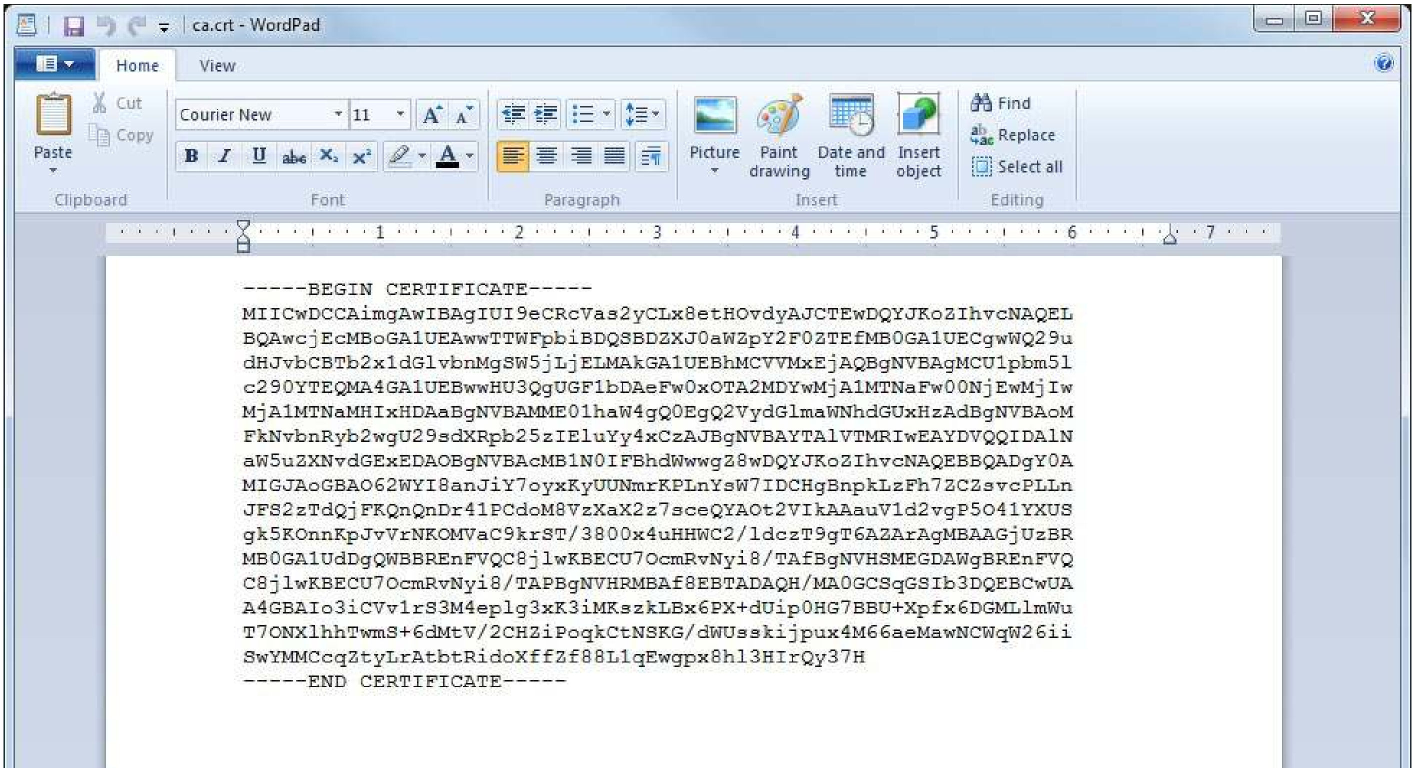
Action: Load XML Config File

Boot configura

The required certificates are as follows, and must use exactly these names.

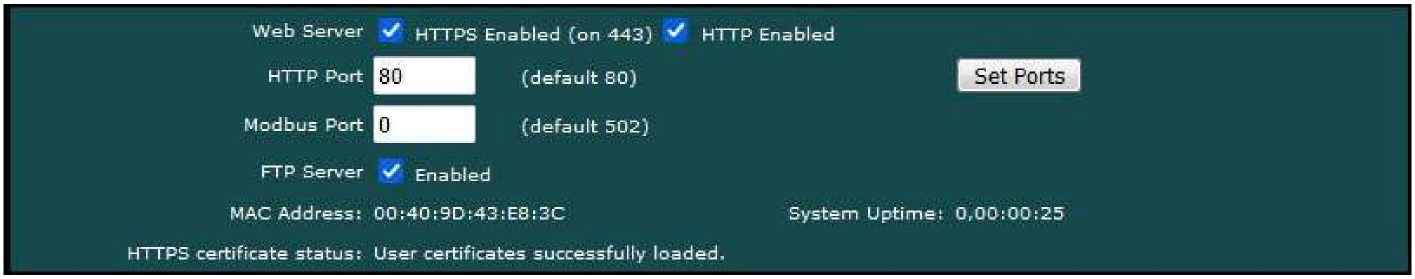
ca.crt	CA Root certificate in PEM format
server.crt	Server certificate in PEM format
server.key	Server private key in PEM format

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).



```
-----BEGIN CERTIFICATE-----
MIICwDCCAimgAwIBAgIUI9eCRcVas2yCLx8etHOvdyAJCTEwDQYJKoZIhvcNAQEL
BQAwcjEcMBoGA1UEAwwTTWFpbiBDQSBDZXJ0aWZpY2F0ZTEfMB0GA1UECgwWQ29u
dHJvbCBTb2x1dGlvbnMgSW5jLjELMAkGA1UEBhMCVVMxEjAQBgNVBAgMCU1pbm5l
c290YTEQMA4GA1UEBwwHU3QgUGF1bDAeFw0xOTA2MDYwMjA1MTNaFw00NjEwMjIw
MjA1MTNaMHIxHDAaBgNVBAMME01haW4gQ0EgQ2VydGlmaWNhdGUxHzAdBgNVBAoM
FkNvbnRyb2wgU29sdXRpb25zIEluYy4xCzAJBgNVBAYTAlVTMRIwEAYDVQQIDAlN
aW5uZXNvdGExEDAOBgNVBAcMB1N0IFBhdWwwgZ8wDQYJKoZIhvcNAQEBBQADgY0A
MIGJAoGBAO62WYI8anJiY7oyxKyUUNmrKPLnYsW7IDCHgBnpkLzFh7ZCZsvcPLLn
JFS2zTdQjFKQnQnDr41PCdoM8VzXaX2z7sceQYAOt2VIkAAauV1d2vgP5O41YXUS
gk5KOnnKpJvVrNKOMVaC9krST/3800x4uHHWC2/ldczT9gT6AZArAgMBAAGjUzBR
MB0GA1UdDgQWBBREnFVQC8jlwKBECU7OcmRvNyi8/TAfBgNVHSMEGDAWgBREnFVQ
C8jlwKBECU7OcmRvNyi8/TAPBgNVHRMBAf8EBTADAQH/MA0GCSqGSIb3DQEBCwUA
A4GBAIo3iCVv1rS3M4eplg3xK3iMKszkLBx6PX+dUip0HG7BBU+Xpfx6DGMLlmWu
T7ONXlhhTwmS+6dMtV/2CHZiPoqkCtNSKG/dWUsskijpux4M66aeMawNCWqW26ii
SwYMMCcqZtyLrAtbtRidoXffZf88L1qEwgpx8hl3HIrQy37H
-----END CERTIFICATE-----
```

If external certificates were loaded successfully, that will be indicated at the bottom of the Network configuration page.



Web Server ☑ HTTPS Enabled (on 443) ☑ HTTP Enabled

HTTP Port 80 (default 80) Set Ports

Modbus Port 0 (default 502)

FTP Server ☑ Enabled

MAC Address: 00:40:9D:43:E8:3C System Uptime: 0,00:00:25

HTTPS certificate status: User certificates successfully loaded.

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 '***********************************************************'
```