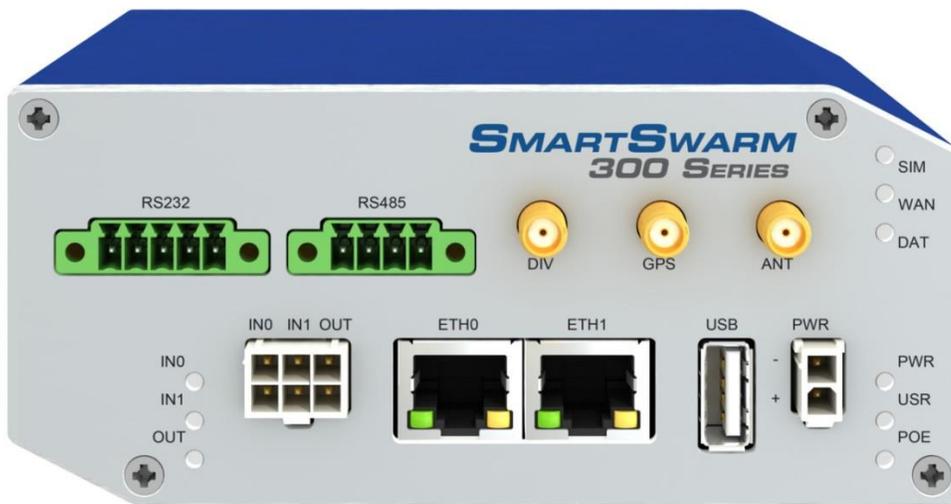


# SmartSwarm 300 Series

## User Manual



**B+B** SMARTWORX

Powered by

**ADVANTECH**

**Advantech B+B SmartWorx Americas**

707 Dayton Road  
Ottawa, IL 61350 USA  
**Phone** (815) 433-5100  
**Fax** (815) 433-5105

**Advantech B+B SmartWorx European Headquarters**

Westlink Commercial Park  
Oranmore, Co. Galway, Ireland  
**Phone** +353 91-792444  
**Fax** +353 91-792445

[www.advantech-bb.com](http://www.advantech-bb.com)  
[support@advantech-bb.com](mailto:support@advantech-bb.com)

**CONTENTS**

List of Tables .....	7
1. Introduction .....	10
1.1 Why Enrich Data? .....	11
1.2 Why Aggregate Data? .....	11
1.3 Why Filter Data? .....	11
1.4 Sampling Theory .....	11
2. Document Structure .....	12
3. Example Workflow.....	15
3.1 Connect Your Hardware .....	15
3.2 Configure Your Device's Connectivity to SmartWorx Hub.....	15
3.3 Configure the Modbus Interface .....	18
3.4 Configure the MQTT interface.....	20
3.5 Build Your Slave Maps .....	21
3.5.1 Discover Your Slaves .....	22
3.5.2 Create/Import Your Slaves.....	23
3.5.3 Export Slave Maps .....	28
3.6 Configure Rules and Topics.....	28
3.7 Verify Your Data Flow .....	31
3.8 Optimize Your System.....	31
4. Connect Your Hardware .....	33
4.1 Mounting the device.....	33
4.1.1 Installing/Removing from a DIN Rail.....	33

4.2 Power Connector PWR .....	34
4.3 Ethernet Port (ETH0 and ETH1) .....	34
4.4 Cellular Connection.....	36
4.4.1 Antenna Connectors ANT, DIV and GPS.....	36
4.4.2 SIM Card Reader .....	37
4.5 RS-232 RS-485 Serial Interface - Connection to Modbus Network.....	38
4.5.1 Wire RS-485 connection .....	38
4.5.2 RS-232 Connection .....	39
4.5.3 Wire RS-485 and RS-422 connection .....	40
4.6 MicroSD Card Reader.....	41
4.7 USB Port.....	41
4.8 I/O Port .....	41
4.9 LEDs .....	41
5. Configure Connectivity to SmartWorx Hub .....	42
5.1 Step 1 - Connect to Local Webserver.....	43
5.2 Step 2 - Configure the Cellular APN details .....	43
5.3 Step 3 - Verify the Secure Connection with SmartWorx Hub .....	43
5.4 Step 4 - Verify That Your Device is Available on SmartWorx Hub .....	43
5.5 Factory Defaults.....	44
6. SmartSwarm 351 on SmartWorx Hub.....	45
6.1 Device Management.....	45
6.2 The Modbus-to-MQTT application .....	46
7. Configure the Modbus Interface .....	47
8. Configure the MQTT interface .....	48
9. Slave Maps and Enrichment .....	53
9.1 Discover .....	56

9.2 Create an Empty Slave Map.....	57
9.3 Import a Slave Map.....	58
9.4 Editing Slaves .....	58
9.4.1 Understanding Your Slave Editor.....	59
9.4.2 Meta Data .....	60
9.4.3 Registers .....	61
9.4.4 Data Types .....	68
9.4.5 Adding Registers .....	70
9.4.6 Editing Registers .....	70
9.4.7 Deleting Registers .....	72
10. Rules and Topics .....	73
10.1 Introduction.....	73
10.2 Events (WHEN).....	75
10.2.1 Read .....	76
10.2.2 Change .....	77
10.2.3 Delta .....	80
10.2.4 High Threshold.....	82
10.2.5 Low Threshold .....	84
10.2.6 High Rate .....	86
10.2.7 Low Rate .....	89
10.2.8 Scheduled .....	91
10.2.9 Global Read.....	92
10.2.10 Global Change.....	92
10.3 Payloads (WHAT) .....	93
10.3.1 Payload Examples .....	94
10.4 Topics (HOW).....	102

10.4.1 Custom Topic Space .....	103
10.4.2 Default Topic Space .....	104
11. Verify your Data Flow .....	107
12. Other Documentation.....	109
13. Appendix 1 - Hardware Ratings .....	109
13.1 Environmental .....	109
13.2 Type Tests .....	110
13.3 Cellular Module .....	110
13.4 Other Technical Parameters .....	111
14. Appendix 2 - General Settings .....	112
14.1 Configurable Items.....	112
14.1.1 Settings .....	112
14.1.2 DHCP .....	113
14.1.3 OpenVPN .....	114
14.1.4 NTP Client .....	117
14.2 Non-Configurable items.....	117
14.2.1 Firewall .....	117
15. Appendix 3 - Diagnostics and Troubleshooting .....	119
15.1 The Local Web Interface .....	119
15.1.1 Home .....	120
15.1.2 Settings .....	120
15.1.3 Troubleshooting.....	120
15.1.4 Hub Client .....	122
15.1.5 Cellular .....	122
15.1.6 Logs.....	123
15.1.7 Modbus.....	123

15.1.8 Debug and Agents.....	124
15.1.9 TSED .....	126
16. Appendix 4 - Slave Map Formats .....	128
16.1 Excel.....	128
16.2 JSON.....	131
17. Appendix 5 - Background Information.....	133
17.1 Modbus Background.....	133
17.2 MQTT Background .....	136
18. Appendix 6 – Dashboards .....	139
18.1 Node-RED.....	139
Advantech B+B SmartWorx Technical Support .....	146

## LIST OF TABLES

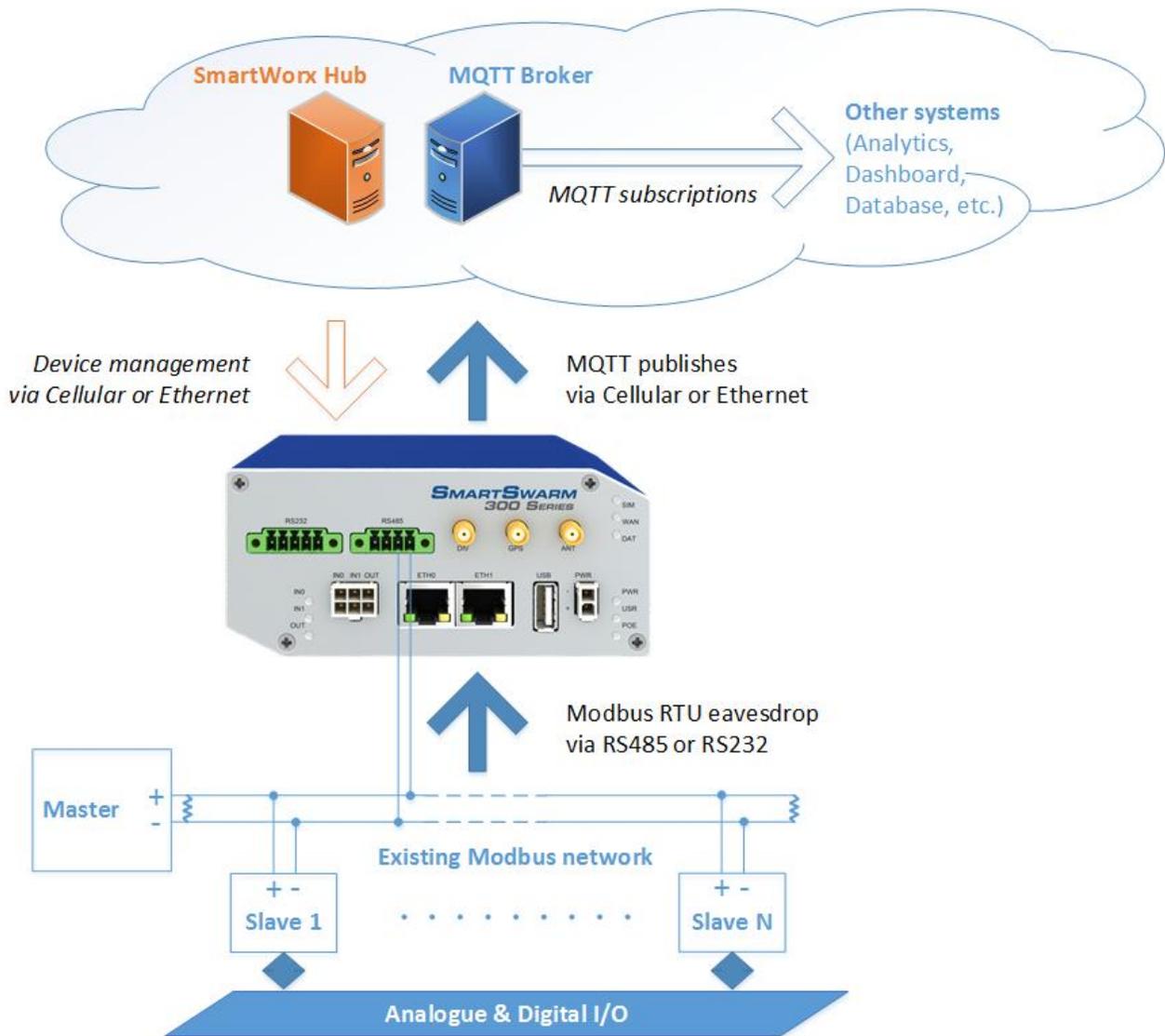
Table 1. Example Modbus Slave Datasheet for Discrete Inputs .....	25
Table 2. Example Excel sheet data derived from Slave Datasheet (Inputs).....	25
Table 3. Example Modbus Slave Datasheet for Input Registers .....	26
Table 4. Example Excel sheet data derived from Slave Datasheet (Input Registers).....	27
Table 5. Example Excel sheet Meta Data.....	27
Table 6. Power connector.....	34
Table 7. Ethernet Ports .....	35
Table 8. Ethernet Port Usage.....	36
Table 9. RS-485 pinout.....	39
Table 10. RS-232 pinout.....	40

Table 11. LED indicators .....	42
Table 12. The Modbus to MQTT application .....	47
Table 13. Modbus Interface.....	48
Table 14. MQTT Interface .....	51
Table 15. MQTT Interface .....	52
Table 16. Slave Map options.....	55
Table 17. Editing Slave Maps .....	59
Table 18. Editing Slave Maps - Rules .....	60
Table 19. Meta Data tab .....	61
Table 20. Register Types .....	61
Table 21. Input Register and Holding Register editable fields.....	65
Table 22. Discrete Input and Coil editable fields .....	68
Table 23. Data Types and Field Values .....	69
Table 24. Rules and Topics fields .....	74
Table 25. Event Types .....	75
Table 26. Events and Data Types: cross-reference .....	76
Table 27. Read Event .....	76
Table 28. Change Event .....	77
Table 29. Delta Event.....	81
Table 30. High Threshold Event.....	82
Table 31. Low Threshold Event.....	84
Table 32. High Rate Event.....	86
Table 33. Low Rate Event .....	90
Table 34. Global Read Event.....	92
Table 35. Global Change Event.....	92
Table 36. Payload options.....	93

Table 37. Event / Payload matrix.....	93
Table 38. The Default Topic .....	104
Table 39. Default Topic example .....	105
Table 40. Verify your Data Flow .....	108
Table 41. Other Documentation.....	109
Table 42. Environmental.....	110
Table 43 Type Tests .....	110
Table 44. Type Tests .....	110
Table 45. Cellular Module.....	111
Table 46. Technical Parameters.....	111
Table 47. OpenVPN fields .....	116
Table 48. Firewall rules.....	118
Table 49. Excel Sheet tabs .....	128
Table 50. Excel sheet, Address tab .....	129
Table 51. Supported Modbus commands.....	134
Table 52. Supported Data Types.....	135
Table 53. Examples for subscribing to different topics in a hierarchical name space .....	137
Table 54. Node Red fields for Gauge node .....	143
Table 55. Node Red fields for Chart node.....	144

1. INTRODUCTION

SmartSwarm 351 is an IoT Gateway appliance powered by B+B SmartWorx SmartSwarm technology. It is intended for use in applications where users need to pass data from legacy Modbus RTU installations into an IoT platform or application, but who can't tolerate any disruption to the Modbus system in order to achieve this.



Using the notion of protocol eavesdropping to non-intrusively extract base data from the messages being sent between the existing master and slave devices in a Modbus network, it leverages the feature-rich data enrichment, filtering and aggregation capabilities of the SmartSwarm software stack to produce event-driven, semantically-searchable, contextualized information, which is passed to the enterprise using the widely-supported MQTT protocol.

### 1.1 WHY ENRICH DATA?

Modbus data is impossible to interpret without very detailed knowledge of the devices producing the data and the sensors connected to them. Anyone looking at Modbus data can see that the value of unit 31, register 40075 is 2397 – but has no way to interpret this data without prior knowledge of its significance. This is not the way that IoT systems operate. One of the core concepts of an IoT architecture is that systems can request information based upon a semantic model – a user can ask for information about temperatures in the rooms of the buildings they manage, and will receive responses in a form which is self-declaring, for example B+B/Ottawa/Conference Room{temperature: 72 degF}. This conversion of raw, unintelligible register values into interpretable information is a fundamental operation in the integration of legacy devices into an IoT architecture.

### 1.2 WHY AGGREGATE DATA?

Modbus is a poll-response protocol. The master device follows a scan pattern which constantly updates an internal database with the most recently recovered data in a particular unit, whether that data has changed or not. Once we start to convert this data into information that is to be sent over, for example, a cellular data link, it becomes important to regulate the flow of data to that which has value. The fact that the temperature in a room is the same as it was five seconds ago is of little value, and we can make significant savings in data transmission and upstream processing costs if we send aggregated data instead -- for example, the max, min and mean temperature each hour.

### 1.3 WHY FILTER DATA?

Aggregating data is fine, of course, but there are certain events that we would want to be informed of on an urgent basis. Examples would include a temperature that has exceeded a threshold, has an excessive rate of change, or has moved by more than a deadband from the last transmitted value. This is the purpose of filtering. A series of event triggers may be configured and the recovered data compared against these triggers with any match resulting in an immediate action.

### 1.4 SAMPLING THEORY

It is important to bear in mind that the SmartSwarm 351 is only eavesdropping on the Modbus network. It cannot influence the Modbus Slaves, or the Modbus Master in any way. Effectively, the SmartSwarm device is two levels removed from the actual process signals in which you may be interested. This is especially important if the original signal is analog.

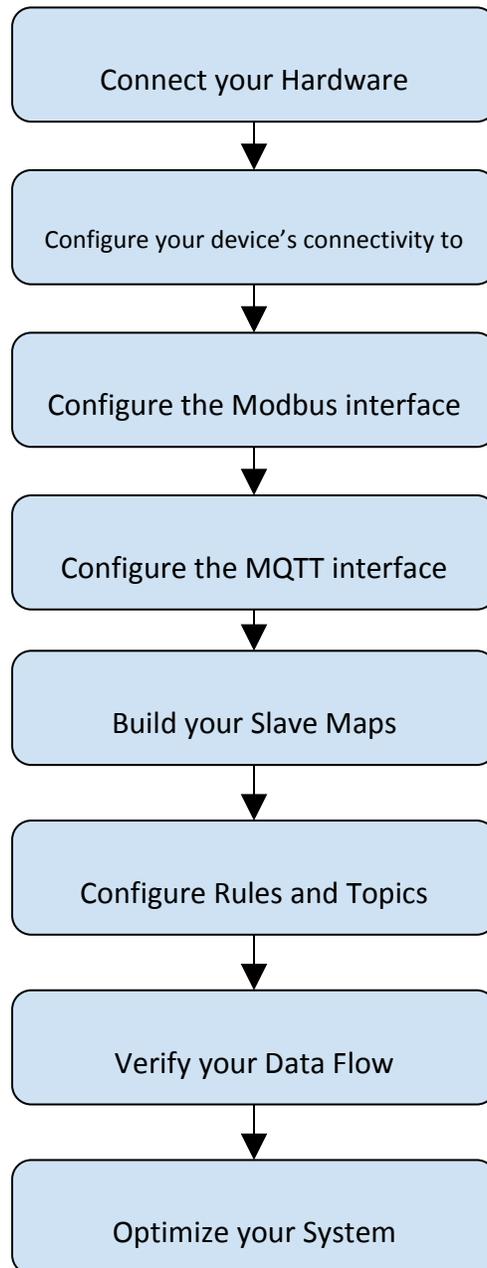
The following example shows a “fast” analog signal that has some high frequency components: The signal is first digitized by a Modbus Slave, at a certain sampling rate, and the quantized values are saved in an Input Register:

This register is then polled by the Modbus Master, at another (slower) sampling rate, and the response values are used by a SCADA system:

The SmartSwarm 351 eavesdrops on the communication between the Modbus Master and Slave. It can only observe the same data that the Modbus Master observes. If the Slave sampling rate and/or the Master sampling rate is not fast enough to capture an event of interest, then that event will be missed.

## 2. DOCUMENT STRUCTURE

This document is organized in accordance with the following flow.





The next chapter walks through an example workflow. This workflow is intended to be an example of how to get your Modbus data publishing to an MQTT server quickly, without getting stuck in the details.

The remaining chapters will provide the necessary details.

### 3. EXAMPLE WORKFLOW

In this section we will walk through an example workflow.

#### 3.1 CONNECT YOUR HARDWARE

First, ensure that your hardware is physically connected.

Connect your antennae to the ANT and DIV connectors.

Insert a valid and data-provisioned SIM card into SIM 1. In this example we will assume that your outbound WAN connection will be using a cellular connection. If this is not the case, and your uplink is solely via Ethernet, then it is not necessary to connect antennae or install a SIM.

In this example, we will connect to an RS-485 Modbus network (this will be the typical configuration).

Physically connect your device to your Modbus network, as per the instructions on the quick-start guide, and as described in the hardware section of this manual.

#### 3.2 CONFIGURE YOUR DEVICE'S CONNECTIVITY TO SMARTWORX HUB

Use an Ethernet cable to connect your local laptop/desktop computer to your SmartSwarm device's ETH0 port. The ETH0 port of the device has IP address 192.168.1.1

The ETH0 port of the device is a DHCP server, so it will automatically serve an IP Address in the 192.168.1.x range to your laptop/desktop computer: please ensure your laptop/desktop computer is configured to accept an IP address automatically from a DHCP server.

Open a web-browser, and browse to 192.168.1.1



### SmartSwarm Local WebServer

This page can be used to configure and diagnostic device's configuration

Below you can find some useful information

Select "Settings"->"Cellular (WAN)", and enter the appropriate APN and network authentication settings for your SIM card. In our example, we only need to enter an APN. Click **EXECUTE**.

## Configure SmartSwarm

Use this page to configure the device to access the hub.

### Cellular (WAN)

APN

Network

Authentication Type:

Network Username

Network Password

PIN Code

Lease Time  
(Seconds)

\*Cellular logs can be found on 'Logs' tab, file /var/log/messages

Execute

That's all you need to do:

The device will now attempt to (a) make a WAN connection using the cellular network; then (b) make a secure connection to SmartWorx Hub (on hub.bb-smartworx.com).

When (a) is successful, the WAN LED will turn on (yellow).

When (b) is successful, the USR LED will turn on (yellow).

The time it takes for (a) to be successful depends on your cellular network, but you should expect it to be successful within minutes.

If the WAN LED is not turning on you may have entered invalid APN or network credential information for that SIM card.

Please verify that you are using a valid SIM card and valid cellular settings.

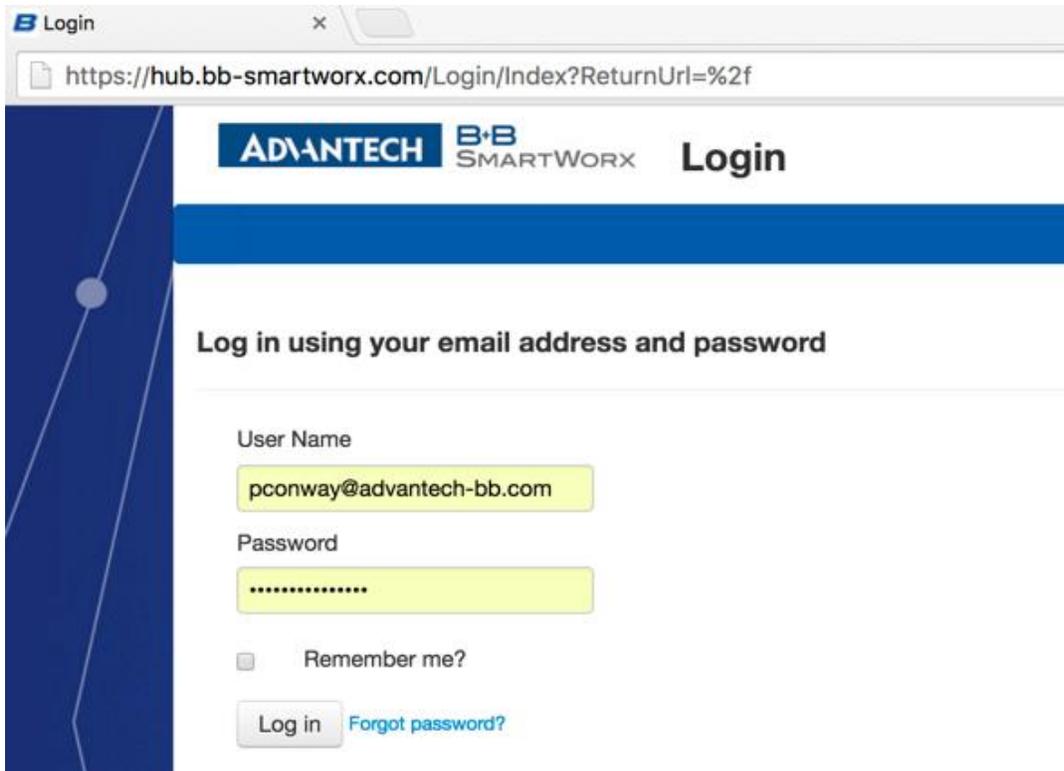
When the USR LED is on (yellow), your device has a secure connection to SmartWorx Hub.

The following graphic shows that the WAN and USR LEDs are both on (yellow), and the RS-485 Modbus network is connected.

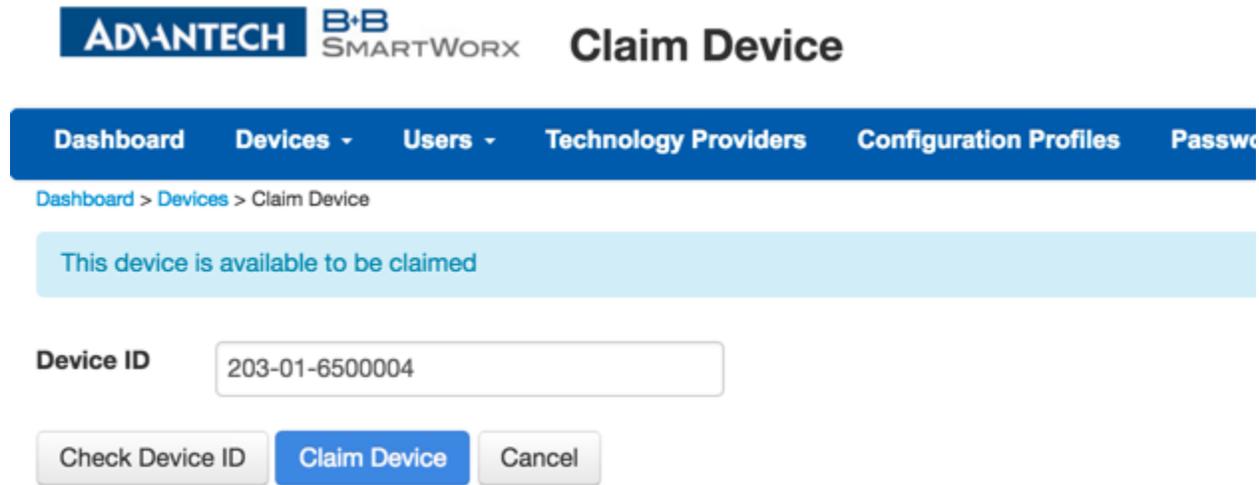


Open a browser page, and login to SmartWorx Hub on <https://hub.bb-smartworx.com>.

In this example, we assume that (a) you have an account to login with SmartWorx Hub, and (b) you are using the cloud instance of SmartWorx Hub to manage your devices.

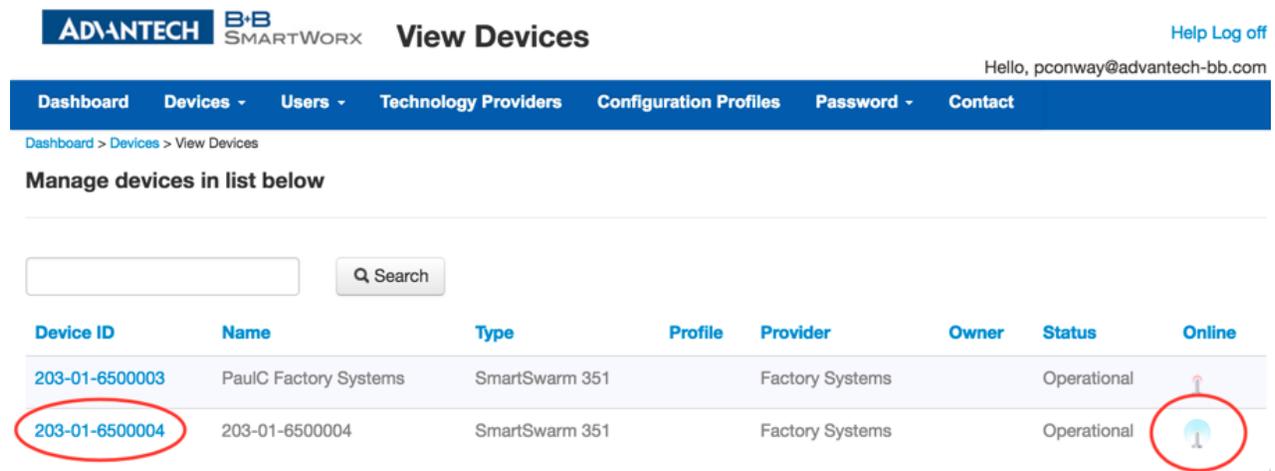


Go to the “Devices”->“Claim Device” screen to bring your new SmartSwarm Device into your Device farm. Type in your Device’s Device-ID (this is written both on the Device itself and on the box that you took your Device out of) and select ‘Check Device ID’ to check that your device is available to be claimed by you. Assuming that it is, you may then select “Claim Device”.



Your Device is now available for you to manage.

By selecting the ‘Devices/View Devices’ screen we can see that the device is available, and that it is currently Online.



### 3.3 CONFIGURE THE MODBUS INTERFACE

Select the Device (click the Device ID link).

Now you can navigate to the Modbus-to-MQTT application and modify the application settings.

Dashboard > Devices > Manage Device

Device ID 203-01-6500004

Name

Status

Firmware



DeviceType SG30300320-51

Online

- Save
- Cancel
- Push Firmware
- History
- Settings
- Add/Upgrade Apps

### Manage Apps

Remove Selected

	Name	Tag	Type	Version	Help	Added
<input type="checkbox"/>	Modbus2MQTT	Modbus2MQTT	Application	0.4.9	<a href="#">Help</a>	6/8/2016 3:14:42 PM

Now configure the Modbus Settings so that they match the actual Modbus configuration of your Modbus network.

Apply the changes.

**Settings**

Dashboard > Devices > Manage Device > Settings

Modbus  
Decoder  
MQTT

**Application Settings**

Device ID: 203-01-6500004  
 Application Name: Modbus2MQTT  
 Version: 0.4.9  
 Tag: Modbus2MQTT

Save Tag Cancel **Apply changes**

\* Required Field

**Modbus**

**Sniffer Settings**

Port:	RS485
Baud Rate:	9600
Parity:	None
Databits:	8
Stopbits:	1
Response Timeout (secs):	1

### 3.4 CONFIGURE THE MQTT INTERFACE

Select MQTT from the list on the left hand pane and configure your MQTT interface.

We assume that you already have an MQTT broker that you can publish to. In our example we know we have an MQTT broker available at 52.51.11.241, using the default port 1883.

Modbus  
Decoder  
MQTT

- Required Field

MQTT

Host:	<input type="text" value="52.51.11.241"/>
Port:	<input type="text" value="1883"/>
Username:	<input type="text"/>
Password:	<input type="password"/>
Client ID:	<input type="text"/>
Timeout (secs):	<input type="text" value="60"/>
Retry Interval (secs):	<input type="text" value="10"/>
Keep Alive (secs):	<input type="text" value="60"/>
Reliability:	<input checked="" type="checkbox"/>
Clean Session:	<input checked="" type="checkbox"/>
Enable TLS:	<input type="text" value="No"/>
Verify Server Cert:	<input type="checkbox"/>
Mutual Authentication:	<input type="checkbox"/>
Server Root CA Cert <span style="float: right;">-</span>	
Client Certificate <span style="float: right;">-</span>	

In this example we are not configuring any security in the MQTT interface. We recommend that you use “no security” only until you have verified your connection and data-flow.

*If you do not apply transport layer security settings, your data will be published to the MQTT server in plaintext.*

*If you chose not to apply security settings now, please remember to do so later.*

Once you are confident that your MQTT connection settings are valid, we recommend that you enable TLS and configure a trusted, secure connection between the SmartSwarm device and your MQTT broker.

Remember to Apply your changes.

### 3.5 BUILD YOUR SLAVE MAPS

There are two main ways for you to build your Slave Maps: “Discover” or “Create”.

Both ways are useful, for different reasons. You can safely mix both methods in building your Slave Maps.

3.5.1 DISCOVER YOUR SLAVES

Using the “Discover” option, you can let the SmartSwarm device self-learn the Modbus slave network. Actually, your SmartSwarm device has been self-learning already. Your device has been physically connected to the Modbus network since the beginning of this workflow. The Modbus configuration settings were deployed to the device in an earlier step.

The Device will continuously learn from the actual Modbus data traffic, even after the initial learning period. As the SmartSwarm device is a passive sniffer device, the time it takes for it to learn all of the Modbus slaves and registers is outside of the control of the device. It depends entirely upon how the Modbus Master is configured. We recommend that you initially leave your device running on the Modbus network for a period of time - maybe 10 or 15 minutes. At this time the device will have learned from the Modbus traffic. It is likely that it won't have learned everything in this amount of time, but it should have learned enough to enable you to carry on to the enrichment steps.

Now go to the “Decoder” screen, and click on “Sync Maps”.

Dashboard > Devices > Manage Device > Settings

Modbus  
**Decoder**  
MQTT

### Application Settings

**Device ID** 203-01-6500004

**Application Name** Modbus2MQTT

**Version** 0.4.9

**Tag**

\* Required Field

Decoder

**Slaves**

- ☰
- Export Maps
- Download Templates
- **Sync Maps**
- New Map
- Load Map

Port	Slave	Name	Description	Location

Refresh your SmartWorx Hub screen to see the discovered devices.

In our example, only one Slave device has been discovered on the Modbus network.

Dashboard > Devices > Manage Device > Settings

Modbus  
Decoder  
MQTT

### Application Settings

Device ID 203-01-6500004

Application Name Modbus2MQTT

Version 0.4.9

Tag

\* Required Field

Slaves					
Port	Slave	Name	Description	Location	
RS485	1	undefined	undefined	undefined	<input type="button" value="-"/> <input type="button" value="Edit"/>

In this case the discovered slave is at address 1. We will refer to this slave as “Slave 1” later.

Although not shown in this example, you can now edit this slave, and enrich the discovered slave data. We will show some enrichment editing of Slave 1 a little later.

Next we will use the “import” method to get more slave maps into the system.

### 3.5.2 CREATE/IMPORT YOUR SLAVES

Using the “**Create**” option you can import a pre-prepared, pre-enriched set of Slave data, in either Excel or JSON format.

We provide a number of aids for you to use the “**Create**” option.

First, you can download templates directly from here using the “Download Templates” option. At the time of writing we have templates in .xls, .xlsx, and .json formats.

Second, you can create a slave and enter all enrichment for it using the “New Map” option.

Third, you can import a pre-prepared, pre-enriched, set of slave data using the “Load Map” option. When importing slave data you must use one of the template formats provided by “**Download Templates**”.

Dashboard > Devices > Manage Device > Settings

Modbus  
Decoder  
MQTT

### Application Settings

Device ID 203-01-6500004

Application Name Modbus2MQTT

Version 0.4.9

Tag

\* Required Field

Decoder

#### Slaves

- Export Maps
- Download Templates
- Sync Maps
- New Map
- Load Map

Port	Slave	Name	Description	Location		
RS485	1	Liebert	Power Monitor	Server Room	-	Edit

In our example we are going to load a pre-prepared slave-map from a template file.

To create the pre-prepared slave-map we took the datasheet of a slave device (an Emerson Liebert nFinity UPS), and we populated the Template accordingly.

Here's an example from the Modbus "Inputs" section of the actual device Datasheet:

# Liebert Nfinity

## Supported Modbus Points

Data Point	Register	Coil	# of Reg.	Scale	Notes/ Units
Automatic Battery Test Enabled	10003	3	1	1	
Battery Charger On	10044		1	1	
Inverter Ready	10047		1	1	
Power Factor Correction State	10050		1	1	
Load On Inverter	10073		1	1	
Bypass Active	10074		1	1	
Replace Battery	10081		1	1	
Battery Under Test	10082		1	1	
Load On Battery	10128		1	1	
Load On Bypass	10129		1	1	

Table 1. Example Modbus Slave Datasheet for Discrete Inputs

We derived the following “Inputs” Excel sheet information from this datasheet, using the provided Excel template.

	A	B	C	D	E	F
1	address	name	alias	val0	val1	
2	2	Automatic Battery Test Enabled	AutomaticBatteryTest	Battery Test Disabled	Battery Test Enabled	
3	43	Battery Charger on	BatteryCharger	Off	On	
4	46	Inverter Ready	Inverter	Not Ready	Ready	
5	49	Power Factor Correction State	PowerFactor	Off	On	
6	72	Load On Inverter	LoadOn	Inverter Off	Inverter On	
7	73	Bypass Active	Bypass	Not Active	Active	
8	80	Replace Battery	ReplaceBattery	Not required	Required	
9	81	Battery Under Test	BattUnderTest	Off	On	
10	127	Load On Battery	LoadOnBatt	Off	On	
11	128	Load On Bypass	LoadOnBypass	Off	On	

Table 2. Example Excel sheet data derived from Slave Datasheet (Inputs)

**NOTE** that the address field in our template is declared as the offset from the base address of the register type. In our example the device uses the 10,xxx range for its Input Status registers. (Other devices may use 10x,xxx.) So the first IS register (10,001) corresponds to offset 0. Hence the device register 10,003 becomes our IS register 2. Here’s an example of the Modbus “Input Registers” section of the datasheet:

Number Of Input Lines	30004		1	1	Bits 12 - 15
Number Of Bypass Lines	30004		1	1	Bits 4 - 7
Number Of Output Lines	30004		1	1	Bits 8 - 11
Number Of Power Mod.	30010		1	1	
Number Of Battery Modules Installed	30011		1	1	
Device Maximum Frame Capacity	30023		2	1	
Device System Capacity	30025		2	1	VA
Nominal Input Voltage	30027		1	1	V
Nominal Output Voltage	30028		1	1	V
Nominal Static Bypass Switch Voltage	30029		1	1	V
Nominal Input Frequency	30031		1	10	Hz
Nominal Output Frequency	30032		1	10	Hz
Nominal Power Factor	30033		1	100	
Nominal Battery Voltage	30034		1	1	V
Auto Restart Delay	30051		1	1	seconds
Device Auto Restart Percent Setpt	30052		1	1	%
Device Low Battery Time	30053		1	1	min
Next Battery Auto Test Time	30057		1	1	minutes
Overload Alarm Limit	30067		2	1	VA
Minimum Redundant Power Modules	30074		1	1	
Load (Apparent Power)	30102		2	1	VA
Load (Real Power)	30104		2	1	W
Load / Capacity	30106		1	1	%
Input Frequency	30107		1	10	Hz
Output Frequency	30108		1	10	Hz
Bypass Frequency	30109		1	10	Hz

Table 3. Example Modbus Slave Datasheet for Input Registers

We derived the following “Input Registers” Excel sheet information from this Datasheet, using the provided Excel Template:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	address	address_offset	name	alias	datatype	length	zero_value	max	min	scaling	units	num	val
2	3	12	Number of Input Lines	NumInputs	ENUM	4					1		
3	3	4	Number of Bypass Line	NumBypass	ENUM	4					1		
4	3	8	Number of Output Line	NumOutput	ENUM	4					1		
5	9	0	Number of Power Mod	NumPowerM	UJINT16						1		
6	10	0	Number of Battery Mod	NumBattMoi	UJINT16						1		
7	22	0	Device Maximum Fram	MaxFrameC	UJINT32						1		
8	24	0	Device System Capacit	DeviceSysC	UJINT32						1 VA		
9	26	0	Nomina Input Voltage	NominalInV	UJINT16						1 V		
10	27	0	Nominal Output Voltag	NominalOut	UJINT16						1 V		
11	28	0	Nominal Static Bypass	NominalStat	UJINT16						1 V		
12	56	0	Next Battery Auto Test	NxtBattTest	UJINT16						1 min		
13	101	0	Load (Apparent Power)	Ld_apparen	UJINT32						1 VA		
14	103	0	Load (Real Power)	Ld_real	UJINT32						1 W		
15	105	0	Load / Capacity	Ld_percent	UJINT16						1 %		
16	106	0	Input Frequency	InFreq	UJINT16						10 HZ		
17	107	0	Output Frequency	OutFreq	UJINT16						10 HZ		
18	108	0	Bypass Frequency	ByFreq	UJINT16						10 HZ		

Table 4. Example Excel sheet data derived from Slave Datasheet (Input Registers)

Next, optionally fill in the “Meta” data in the Excel sheet:

	A	B	C	D	E	F	G	H
1	description	installation_date	location	manufacturer	name	product_code	value_byte_order	version
2	UPS	14-Mar-16	Thomond/Datacentre1	Emerson	nFinity UPS	Liebert nFinity	No Swap	1.00
3								
4								
5								

Table 5. Example Excel sheet Meta Data

Now, we’re going to import this pre-prepared Excel file into SmartWorx Hub.

We click on “Load Map”, then browse to the prepared Excel file. Select the file, and the Slave Map will be immediately loaded.

Dashboard > Devices > Manage Device > Settings

Modbus  
Decoder  
MQTT

Application Settings

Device ID 203-01-6500004

Application Name Modbus2MQTT

Version 0.4.9

Tag

Save Tag Cancel **Apply changes**

\* Required Field

Decoder

Slaves

Port	Slave	Name	Description	Location	
RS485	1	Liebert	Power Monitor	Server Room	- Edit
RS485	2	nFinity UPS	UPS	Thomond/Datacentre1	- Edit

Don't forget to Apply Changes.

3.5.3 EXPORT SLAVE MAPS

Once you have Slave Maps on your system - whether you created them manually, imported them, or discovered them and subsequently enriched them - you can then export them.

This is very useful if you have many Slave Devices of the same type. You can create (and enrich) one Slave, then export it to a .json format file.

You can edit the file offline (for example, to change the Meta Data, and Slave Address), then re-import it as a new slave.

All previously applied enrichment will be available immediately on your new slave.

When you use the export utility all of your existing slave maps will be exported into a single archive file (.zip), which goes directly into the "downloads" folder defined by your browser.

3.6 CONFIGURE RULES AND TOPICS

Click on **Edit** to enrich a slave, and to apply Events, Rules and Topics.

We will now enrich the data in Slave 1.

First, add some Metadata enrichment. Remember to **Save** when you're happy with the enrichment.

Dashboard > Devices > Manage Device > Settings > Slave

Save Push to Device Exit Editor

Meta Inputs (1x) Coils (0x) Input Registers (3x) Holding Registers (4x) Rules and Topics

Description	Install Date	Location	Manufacturer	Name	Product Code	Byte Order	Version
Power Monitor	30 Jun 2016	Server Room	Emerson	Liebert	PowerSure	No Swap	undefined

In our example there were some Input Registers discovered on Slave 1 during the “discovery” stage. Go to the **Input Registers** tab, and enrich the registers.

Dashboard > Devices > Manage Device > Settings > Slave

Save Push to Device Exit Editor

Meta Inputs (1x) Coils (0x) Input Registers (3x) Holding Registers (4x) Rules and Topics

Address	Bit Offset	Name	Alias	Data Type	Width	Zero Value	Max	Min	Scaling	Units	Num Value	Str Value	
29		Nominal Input Current	InputCurrent	UINT16	16				1	A			- +
30		Nominal Input Frequency	InputFreq	UINT16	16				10	Hz			- +
31		Nominal Output Frequency	OutputFreq	UINT16	16				10	Hz			- +
32		Nominal Power Factor	PowerFactor	UINT16	16				100				- +
33		Nominal Battery Voltage	BattVoltage	UINT16	16				1	V			- +
34		undefined	undefined	UINT16	16					undefined			- +

In our example we’re enriching the slave’s input register data from the slave’s datasheet (an Emerson power monitoring UPS).

Don’t forget to “**Save**” as you go.

Now, we’re ready to apply some Rules and Topics. Select the **Rules and Topics** tab.

The first thing we notice is that the enrichment data we have previously provided has already been applied to this panel.

Dashboard > Devices > Manage Device > Settings > Slave

Exit Editor

Meta Inputs (1x) Coils (0x) Input Registers (3x) Holding Registers (4x) Rules and Topics

Save Rules Push Rules Register 29 IR A

Address	Type	Name	Units	Event	Payload	QOS	Retain	MQTT Topic	Default Topic	
29	IR	Nominal Input Current	A	None	Default	Exactly Once	<input type="checkbox"/>	Server_Room/Power_Monitor/Lieb...	✓	- +
30	IR	Nominal Input Frequency	Hz	None	Default	Exactly Once	<input type="checkbox"/>	Server_Room/Power_Monitor/Lieb...	✓	- +
31	IR	Nominal Output Frequency	Hz	None	Default	Exactly Once	<input type="checkbox"/>	Server_Room/Power_Monitor/Lieb...	✓	- +
32	IR	Nominal Power Factor		None	Default	Exactly Once	<input type="checkbox"/>	Server_Room/Power_Monitor/Lieb...	✓	- +
33	IR	Nominal Battery Voltage	V	None	Default	Exactly Once	<input type="checkbox"/>	Server_Room/Power_Monitor/Lieb...	✓	- +
34	IR			None	Default	Exactly Once	<input type="checkbox"/>	Server_Room/Power_Monitor/Lieb...	✓	- +

There's a line entry in this table for every discovered register on this Slave (even for the non-enriched registers). The MQTT Topic has taken custom values, which are derived from the Meta data. You may leave these as-is, or you may override the custom-defaults and define an MQTT topic for every register-rule.

You define an event ("when" a matching data pattern occurs on the Modbus network) in the "Event" column.

You define the payload that will be published ("what" is published when an event occurs) in the "Payload" column.

You define the MQTT topic the payload will be published on ("how" the payload is published when an event occurs) in the "MQTT Topic" and "Default Topic" columns.

Dashboard > Devices > Manage Device > Settings > Slave

Exit Editor

Meta Inputs (1x) Coils (0x) Input Registers (3x) Holding Registers (4x) Rules and Topics

Save Rules Push Rules Register 30 IR Hz

Change by: 1 percent

Address	Type	Name	Units	Event	Payload	QOS	Retain	MQTT Topic	Default Topic	
29	IR	Nominal Input Current	A	None	Default	Exactly Once	<input type="checkbox"/>	Server_Room/Power_Monitor/Lieb...	✓	- +
30	IR	Nominal Input Frequency	Hz	Change	Default	Exactly Once	<input type="checkbox"/>	Server_Room/Power_Monitor/Lieb...	✓	- +
31	IR	Nominal Output Frequency	Hz	None	Default	Exactly Once	<input type="checkbox"/>	Server_Room/Power_Monitor/Lieb...	✓	- +
32	IR	Nominal Power Factor		None	Default	Exactly Once	<input type="checkbox"/>	Server_Room/Power_Monitor/Lieb...	✓	- +
33	IR	Nominal Battery Voltage	V	None	Default	Exactly Once	<input type="checkbox"/>	Server_Room/Power_Monitor/Lieb...	✓	- +
34	IR			None	Default	Exactly Once	<input type="checkbox"/>	Server_Room/Power_Monitor/Lieb...	✓	- +

In our example we have created only one Event.

We want to know when the Nominal Input Frequency data value changes by 1 percent.

When that change-event occurs, publish the default payload data - that is, publish the Nominal Input Frequency value on this Input Register for this Slave Device.

Publish using the defined MQTT Topic “Server\_Room/Power\_Monitor/Liebert”, but do **not** also publish on the “Default Topic” (see chapter 10).

**Save Rules next:** This saves your enrichment and rules to the SmartWorx Hub database.

**Push Rules now:** This applies the entire enrichment, including the defined event rules, to the device.

Your device will now apply the Events rules you have enabled, and it will start filtering the data and publishing data in accordance with the rules that you have defined.

In our example the device will publish the Nominal Input Frequency register value to the MQTT server at address 52.51.11.241, using the MQTT Topic “Server\_Room/Power\_Monitor/Liebert”, but only when the value of this register has changed by 1% since the last time it was polled on the Modbus network.

### 3.7 VERIFY YOUR DATA FLOW

The checkpoints for your data flow are:

1. You have a good physical connection between the SmartSwarm device and your Modbus network.
2. A secure connection has been established between your SmartSwarm device and SmartWorx Hub.
3. You have correctly configured the Modbus interface on your SmartSwarm device to correspond with your Modbus network.
  - a. And you have applied your settings to the Device.
4. You have correctly configured the MQTT interface on your SmartSwarm device so that your device can establish a connection with a valid MQTT broker.
  - a. And you have applied your settings to the Device.
5. You have created an Event that will actually trigger, based on matching actual data-conditions on your Modbus network to the Event rule you have created.
  - a. And you have saved your enrichment.
  - b. And you have pushed your enrichment to the Device.

When you are satisfied that you have verified all of these checkpoints, it’s a good time to go back over your setup and apply some optimizations.

### 3.8 OPTIMIZE YOUR SYSTEM

Some suggestions for optimizing your system are:

- **Revisit your MQTT configuration settings, and enable security.**

You have many security options. You will need to read the detailed chapter in this manual regarding MQTT configuration.

We recommend that you also read some generally available MQTT security documentation from the OASIS pages ([www.mqtt.org](http://www.mqtt.org)).

It is important that the MQTT broker you chose to use supports the level of security that you wish to apply.
- **Revisit your Slave Map enrichment pages.**

You are free to enrich your slaves and slave-registers in any way you find most appropriate to your usage. It’s easy to change and re-apply your enrichment settings at any time.

Feel free to prototype different enrichment scenarios until you find one that suits your need.

We recommend that you choose appropriate MQTT topics for your Slave registers, which enable ease-of-use for your solution domain. By defining your Topics carefully, you can enable ease-of-consumption of the data.

In some cases, you will want to consume the data on dashboards.

See Appendix 6 in this manual for an example of how to create your own dashboards using a server-side Node RED service.

- **Revisit your Slave Map Rules**

The data that will be published to your MQTT broker is entirely dependent on the rules that you create. We recommend that you do **not** enable a Read Rule for every register. The IoT environment for consuming data is not intended to be a real-time replica of your Modbus control network. Publishing every read to every register is not only extremely data-intensive across the entire system, it will also have real cost associated with it. We recommend that you do **not** enable a Read Rule for every register. The IoT environment for consuming data is not intended to be a real-time replica of your Modbus control network. Publishing every read to every register is not only extremely data-intensive across the entire system, it will also have real cost associated with it (e.g. data usage on cellular networks).

We recommend that you **do** enable appropriate Event Rules for any interesting or non-normal conditions that you want to keep track of, or be notified about.

We recommend that you **do** enable appropriate Schedule Rules for any data that you wish to monitor on a regular basis.

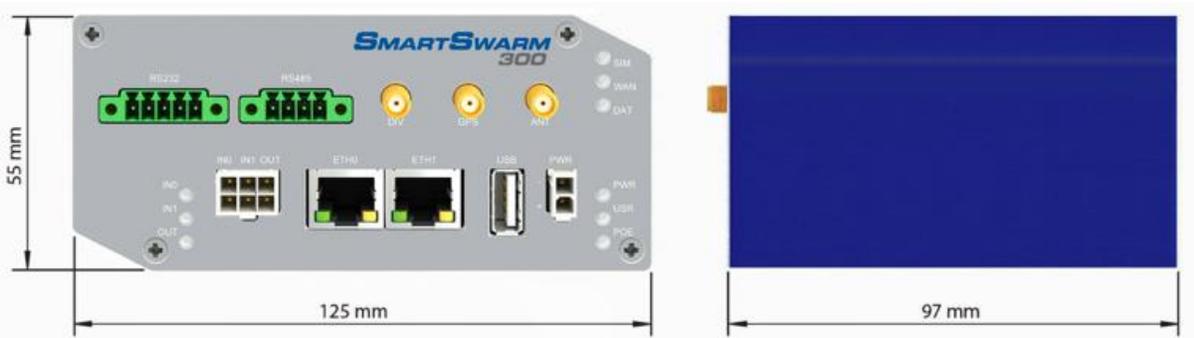
Again, it's best to prototype different scenarios until you find one that suits your need.

It's very easy to change and modify rules at any time, and to re-apply them in run-time without any impact to your system.

**4. CONNECT YOUR HARDWARE**

**4.1 MOUNTING THE DEVICE**

The unit may be mounted in any orientation. It can be simply placed on a flat surface, or it can be DIN rail mounted using the supplied CKD2 holder.



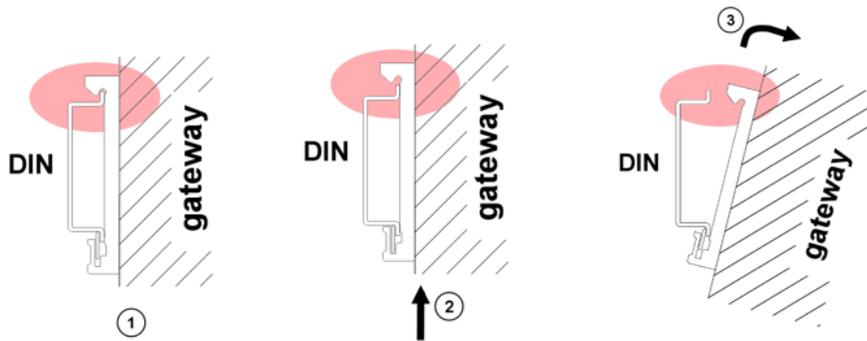
**4.1.1 INSTALLING/REMOVING FROM A DIN RAIL**

The CKD2 holder, which is used for mounting the gateway on a DIN rail, should be mounted such that the smaller flange on the holder is at the top when the unit is mounted on a DIN rail.

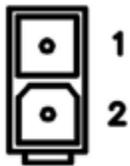


Default orientation of the CKD2 holder

To insert into a DIN rail, hook the lower (longer) flange into the DIN rail then rotate the top of the unit towards the DIN rail until it clicks into place. To remove from the DIN rail, lightly push the IoT gateway upwards until the top part of the CKD2 holder clears the top of the DIN rail. The top of the gateway can then be pulled away from the DIN rail, which will in turn release the lower DIN connection point.



4.2 POWER CONNECTOR PWR



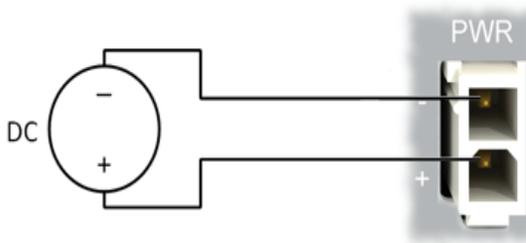
Panel socket 2-pin

Pin	Ident	Description
1	GND(-)	Negative pole of DC supply voltage
2	VCC(+)	Positive pole of DC supply voltage (+10 to +60 V DC)

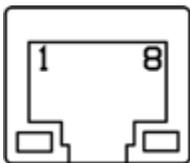
Table 6. Power connector

The unit accepts the connection of power supplies in the range +10 V to +60 V DC. Protection against reverse polarity connection is built into the device.

Circuit example:



4.3 ETHERNET PORT (ETH0 AND ETH1)



Panel socket RJ45.

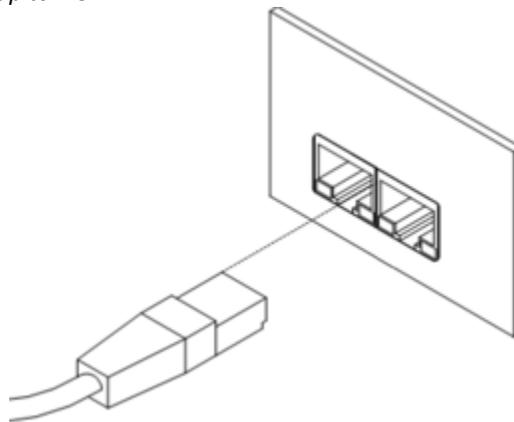
PIN	Signal Mark	Description	Data Flow Direction
1	TXD+	Transmit Data – positive pole	Input/Output
2	TXD-	Transmit Data – negative pole	Input/Output
3	RXD+	Receive Data – positive pole	Input/Output
4	—	—	—
5	—	—	—
6	RXD-	Receive Data – negative pole	Input/Output
7	—	—	—
8	—	—	—

Table 7. Ethernet Ports

Ethernet cables plug directly into the sockets. Always use a cable with an operational locking tab to avoid intermittent communications problems.



*The insulation strength is up to 1.5 kV.*



By default, ETH0 is set up as a DHCP server and is intended for the connection of diagnostic devices. ETH1 is set up as a DHCP client, and may be used as an uplink for MQTT data being sent from the device.

Connector	Purpose	Default Setting
<b>ETH0</b>	LAN port (default) Connect your laptop or PC to this port to get a local web-server for device configuration and diagnostics.	DHCP Server IP Address: 192.168.1.1 NetMask: 255.255.255.0
<b>ETH1</b>	WAN port (default) Connect this port to your WAN to allow the device to obtain access to the remote device management service, SmartWorx Hub, over Ethernet.	DHCP Client The device will automatically obtain an IP address from your DHCP server, if you have a DHCP server provisioned to supply one.

Table 8. Ethernet Port Usage

If a connection exists via ETH1, it will take priority over a cellular connection for northbound data.

## 4.4 CELLULAR CONNECTION

If your device is cellular-enabled, you will need to attach the relevant antennae and install a data-enabled SIM card before you can use cellular connections.

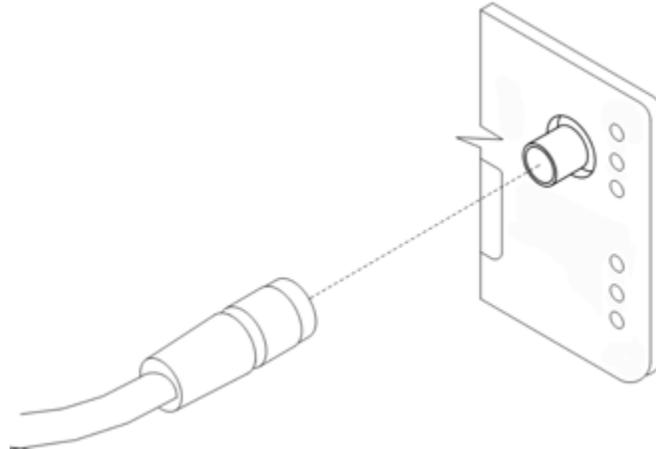
### 4.4.1 ANTENNA CONNECTORS ANT, DIV AND GPS

If cellular communications are required, main and diversity antennas must be connected to the IoT Gateway via SMA connectors on the front panel. The *ANT* connector is used to connect the main antenna of the device. A second, diversity antenna, should be connected to the second cellular antenna connector (*DIV*) in order to improve the gateway radio performance at low signal strength, or in areas where the RF environment is constantly changing. (For example, near a road.) The third connector (*GPS*) is intended for GPS antenna connection and is not currently used by the SmartSwarm 351.



*The device cannot connect to cellular networks without an appropriate antenna connected to ANT*

Antennae are connected by screwing to the SMA connector on the front panel of the IoT Gateway.



#### 4.4.2 SIM CARD READER

Two SIM card readers for 3 V and 1.8 V SIM cards are placed on the rear panel of the device. Only the first of these (SIM1) is currently supported by SmartSwarm 351. In order to operate on a cellular network it is necessary to insert an activated, data enabled SIM card with an unblocked PIN code.

##### 4.4.2.1 INSERTING/REPLACING A SIM CARD:



*Before inserting or removing the SIM card disconnect the device from power supply*

Using a plastic opening tool, or your fingernail, press the SIM card into its slot until you hear a click.

To remove a SIM card press the SIM into the unit until you hear a click. After the click, release the card and it will pop out of its slot.

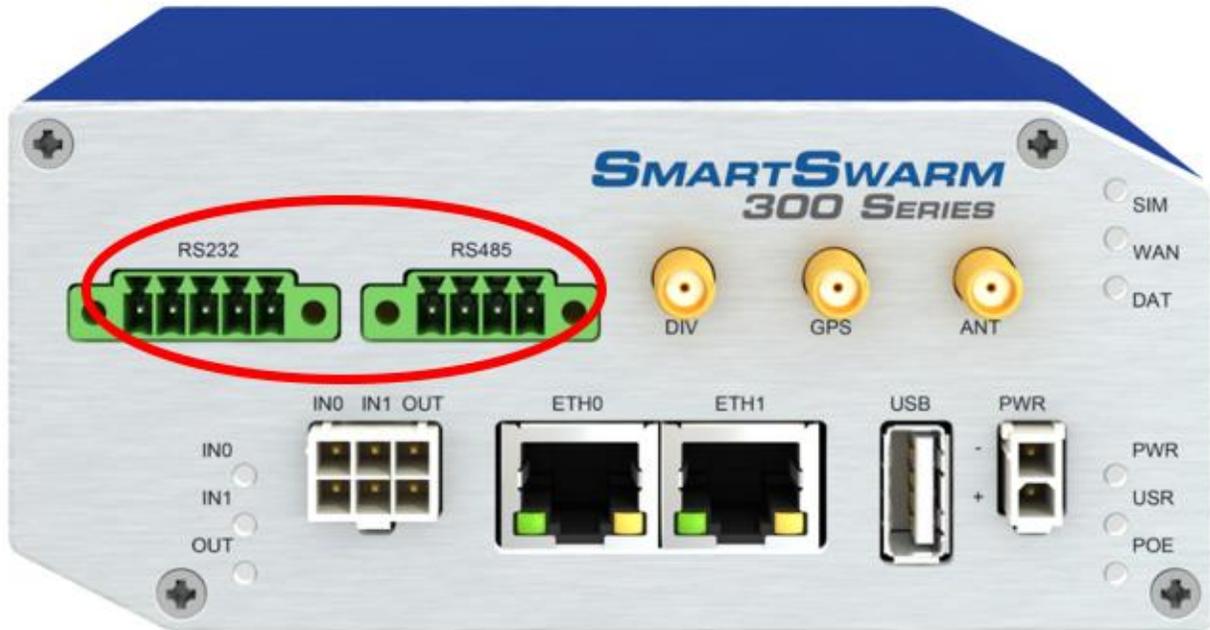
Remove the SIM card and push any other SIM card into the slot until it clicks in place.



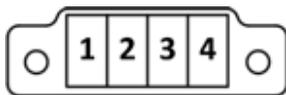
*Only SIM1 is supported in the initial release of SmartSwarm 351*

4.5 RS-232 RS-485 SERIAL INTERFACE - CONNECTION TO MODBUS NETWORK

These interfaces are physically connected on five-pin and four-pin terminal block connectors on the front panel. The insulation strength is up to 2.5 kV. **Attention, connectors are not isolated from each other and share a common ground pin.** The RS-485 ground connection should be made to the RS-232 GND pin.



4.5.1 WIRE RS-485 CONNECTION



RS-485 connector (4 pin)

RS-485 – Pinout			
Pin	Signal	Description	Direction
1	Tx-	Transmit - (Do not connect)	Output

2	Tx+	Transmit + (Do not connect)	Output
3	Rx-	Receive -	Input
4	Rx+	Receive +	Input

Table 9. RS-485 pinout



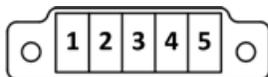
NOTE: this device will only operate in Passive (receive-only) mode. It is not possible to configure it as a Modbus Master, or for it to transmit on the Modbus network.



The RS-485 port provides a transmitter and a receiver. If you connect the transmitter to your Modbus network, you risk interfering with the Modbus communication should the transmitter ever become enabled by software. On this device, the transmitter is not enabled. Nevertheless, we recommend that you do not connect pins 1 and 2.

CONNECT ONLY TO THE RECEIVER, AS SHOWN ABOVE.

#### 4.5.2 RS-232 CONNECTION



RS-232 connector (5 pin)

PIN	Signal	Direction
1	CTS	Output

2	RTS	Input
3	GND	—
4	RXD	Input
5	TXD	Output

**Table 10. RS-232 pinout**

In order to operate on RS-232 based Modbus RTU networks, it is necessary to arrange for the SmartSwarm 351 to receive information from both the RS-232 Transmit and Receive signals, in order that it can monitor both the Modbus commands and the corresponding replies.

RS-232 data taps, B+B part number 9PCDT (9 pin) or 25PCDT (25 pin) are available for this purpose. The following description assumes the use of the 9 pin version.

The device should be inserted into the RS-232 communications line between the Modbus master and slave. Switches 1 & 2 should both be ON in order to pass both Rx and Tx data. In addition, switch 3 should be ON and switch 4 OFF. Connections should then be made between the data tap pin 5 and the SmartSwarm RS-232 connector pin 3, and the data tap pin 2 and the SmartSwarm RS-232 connector pin 4. No other connections are necessary.

The following image shows the correct configuration of DIP switches:



**SmartSwarm 351**

### 4.5.3 WIRE RS-485 AND RS-422 CONNECTION

Please read Section 3.3.3.2 of the “Modbus Serial Line Protocol and Implementation Guide” (Modbus\_over\_serial\_line\_V1\_02.pdf), which covers “Compatibility between 4-Wire and 2-Wire cabling” (see [http://www.modbus.org/docs/Modbus\\_over\\_serial\\_line\\_V1\\_02.pdf](http://www.modbus.org/docs/Modbus_over_serial_line_V1_02.pdf)). The RS-485 interface of the SmartSwarm 351 is effectively a 2-wire interface. In order to connect a 4-wire system to the device, you have to short the Transmit and Receive pairs together.



In some situations this is not an issue, but there are some risks with this approach.

- The SCADA master will see a repeat of its command, followed by the slave reply, and this combination may be rejected as invalid depending upon the characteristics of the SCADA master driver.
- Similarly, the Modbus slave will see its own reply and may try to interpret this as a new command and subsequently fail to recognize a valid command being sent by the host.

For this reason, it is strongly recommended that if an alternative tap-in point exists which operates at either RS-232 or 2-wire RS-485 levels, this should be used.

If there is no alternative to tapping into a 4-wire circuit, then the safest way is to use an RS-232 data tap as outlined above, and connect an RS-232 to RS-422 (or 4-wire RS-485 as appropriate) converter on either side of the data tap. Advantech B+B SmartWorx offers a range of such converters. Please determine the most suitable combination for your application.

**4.6 MICROSD CARD READER**

The MicroSD card socket, located on the rear panel of the unit, is currently unused by SmartSwarm 351

**4.7 USB PORT**

The USB port, located on the front panel, is currently unused by SmartSwarm 351.

**4.8 I/O PORT**

The I/O port, located on the front panel, is currently unused by SmartSwarm 351.

**4.9 LEDS**

The following table describes the LED operation on the SmartSwarm device

LED	Color	State	Description
PWR	Green	Off	No power
		On	Device is booting
		Blinking	Device is in normal operating mode
		Fast Blinking	Device is updating firmware. Do not power off
USR	Yellow	Off	The device does not have a working session established with SmartWorx Hub
		On	The device has a working secure session established with SmartWorx Hub
PoE	Not Used	Not Used	Not Used
DAT	Red	Off	There is no communication on the cellular interface at this moment
		Blinking	There is communication in progress on the cellular interface

SIM	Green	Off	Reset button pressed or the device is booting
		On	Ready for operation. SIM 1 is enabled
WAN	Yellow	Off	There is no cellular connection between the SmartSwarm device and the cellular service provider
		On	A cellular connection has been successfully established between the SmartSwarm device and the cellular service provider
INO	Green	Off	The default state
		On	Binary input no. 0 is active (user defined)
IN1	Green	Off	The default state
		On	Binary input no. 1 is active (user defined)
Out	Yellow	Off	The default state
		On	Binary output is active (user defined)
ETH0	Green	On	10 Mb/s
		Off	100 Mb/s
ETH1	Yellow	On	The network cable is connected
		Off	Network cable is not connected
ETH1		Blinking	Data transmission in progress

Table 11. LED indicators

**5. CONFIGURE CONNECTIVITY TO SMARTWORX HUB**

All major configuration of the SmartSwarm 351 is performed using the SmartWorx Hub cloud based management platform. If you do not already have a SmartWorx Hub account, please contact your local B+B representative to arrange for one to be set up. You will need to provide the following information in order for an account to be set up:

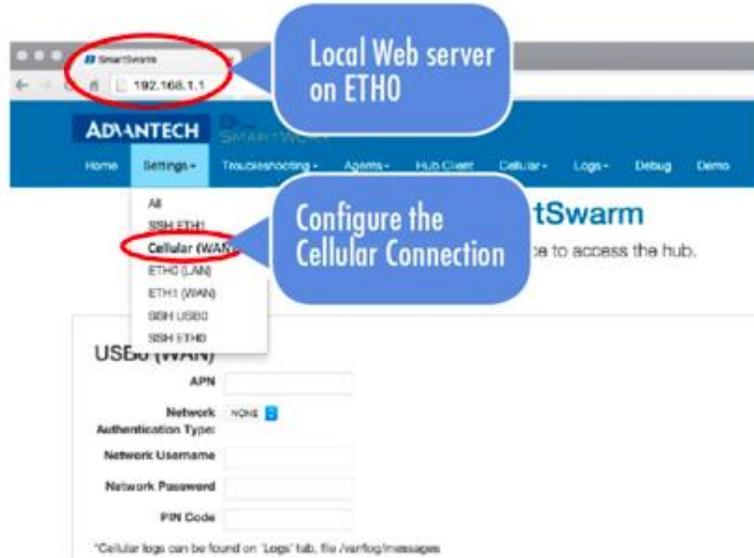
- An Administrator contact name and email address
- The Device ID of the SmartSwarm 351 devices you already have taken delivery of (from the SmartSwarm 351 product label on each device)
- The MAC ID of the first Ethernet port each device (from the SmartSwarm 351 product label on the devices)

	<p><i>SmartWorx Hub is accessed via the primary uplink port on the SmartSwarm 351. This is ETH1 if it is connected to a local LAN providing outbound (internet) access, or the cellular connection if no outbound LAN connection exists.</i></p> <p><i>The connection status to SmartWorx Hub is indicated by the LEDs on the front panel. The USR LED will be solid ON (yellow) if a secure connection to SmartWorx Hub has been achieved.</i></p> <p><i>Please refer to the 'Verification' section below for further details on how to confirm this connection status.</i></p>
---	--

If the Internet connection is to be via cellular connection, ensure that appropriate antennae and SIM cards are inserted before moving on to the first step below.

**5.1 STEP 1 - CONNECT TO LOCAL WEBSERVER**

Connect a local laptop or desktop PC to ETH0. Open a browser and navigate to 192.168.1.1. Note that if you have another LAN connection (e.g. via Wi-Fi) you may need to disconnect this second session, depending upon your network settings and the domain of the LAN.



**5.2 STEP 2 - CONFIGURE THE CELLULAR APN DETAILS**

Enter the APN name and optional credentials as required by your SIM card provider / network operator. Apply it. The WAN LED will turn ON (yellow) when the cellular connection has been successfully established.

**5.3 STEP 3 - VERIFY THE SECURE CONNECTION WITH SMARTWORX HUB**

The USR LED will turn on (yellow) when the device successfully makes a secure connection with SmartWorx Hub (<https://hub.bb-smartworx.com>).

**5.4 STEP 4 - VERIFY THAT YOUR DEVICE IS AVAILABLE ON SMARTWORX HUB**

In order to verify the installation, and to ensure that you have correctly claimed the device within your SmartWorx Hub account, please confirm that the device is shown as “Online” in SmartWorx Hub (For further information on SmartWorx Hub, please refer to the SmartWorx Hub user manual).



If a configuration for the device has already been created in SmartWorx Hub it will be automatically downloaded to the device during this first connection.

### 5.5 FACTORY DEFAULTS

If the unit is not connecting as expected it may be reset to Factory Defaults at any time by pressing the Reset button on the back-panel of the device for more than 10 seconds.

6. SMARTSWARM 351 ON SMARTWORX HUB



Once you have Claimed your Device on SmartWorx Hub you may edit and configure it. If your device is currently offline, all changes you make are queued. All of your changes will be immediately applied as soon as the device comes online.

6.1 DEVICE MANAGEMENT

Please refer to the SmartWorx Hub user manual for more detailed information on general Device Management.

Find the device that you wish to manage in the “View Devices” screen, and click on it to open the “Manage Device” screen.

Dashboard > Devices > View Devices

Manage devices in list below



Device ID	Name	Type	Profile	Provider	Owner	Status	Online
203-01-1000102	203-01-1000102	V3	CP1	Technology Provider A		Operational	
203-01-1000111	203-01-1000111	V3	GO1	Technology Provider A	Gerry Owner	Operational	
203-01-1000112	203-01-1000112	V3				Operational	
203-01-1000113	203-01-1000113	V3				Operational	
203-01-299999	203-01-299999	V2				Operational	
203-01-5100342	203-01-5100342	V2				Operational	
203-01-5321999	203-01-5321999	V2				Operational	
203-01-6200183	203-01-6200183	V3				Operational	
203-01-7000000	203-01-7000000	Smart swarm 300 v2.0		Technology Provider A		Operational	
203-01-7000001	203-01-7000001	Smart swarm 300 v2.0				Operational	

Dashboard > Devices > Manage Device

Device ID 203-01-6500003

Name

Status

Firmware

DeviceType SG30300322-51

Online

[Save](#) [Cancel](#) [Push Firmware](#) [History](#) [Settings](#) [Add/Upgrade Apps](#)



Manage Apps

[Remove Selected](#)

Name	Tag	Type	Version	Help	Added
<input type="checkbox"/> Modbus2MQTT	Modbus2MQTT	Application	0.4.9	<a href="#">Help</a>	5/24/2016 11:39:29 AM

[1](#)

From here you may select the name of the application (“Modbus-to-MQTT”) in order to configure the Modbus-to-MQTT Application.

6.2 THE MODBUS-TO-MQTT APPLICATION

The settings for the Modbus-to-MQTT application are split into 3 functional areas:

Functional Area	Description	Reference
<b>Modbus</b>	Configure the Modbus interface of the SmartSwarm 351. This configuration must match the configuration of the Modbus network you’re monitoring.	See chapter 7.
<b>MQTT</b>	Configure the MQTT interface of the SmartSwarm 351.	See chapter 8.
<b>Decoder</b>	This section enables you to <b>Build</b> and <b>Enrich</b> the Slave Maps that are of interest to you. This section also enables you to define the Rules you wish to create to publish data on the MQTT interface, and the MQTT topics that you will publish that data on.	See chapter 9 for Building and Enriching Slave Maps. See chapter 10 for defining Rules and Topics.

Table 12. The Modbus to MQTT application

## 7. CONFIGURE THE MODBUS INTERFACE

Dashboard > Devices > Manage Device > Settings

Modbus  
Decoder  
MQTT

### Application Settings

Device ID 203-01-6500003

Application Name Modbus2MQTT

Version 0.4.9

Tag

\* Required Field

Modbus

#### Sniffer Settings

Port:	<input type="text" value="RS485"/>
Baud Rate:	<input type="text" value="9600"/>
Parity:	<input type="text" value="None"/>
Databits:	<input type="text" value="8"/>
Stopbits:	<input type="text" value="2"/>
Response Timeout (secs):	<input type="text" value="1"/>

When configuring the Modbus interface, please ensure that you match the Modbus Master configuration on the Modbus network you are monitoring.

Setting	Description
Port	RS-232 or RS-485. The SmartSwarm 351 can only 'sniff' on one of these physical ports at a time.
Baud Rate	1200 through 115200
Parity	None, Even or Odd
Databits	8
Stopbits	1 or 2

<b>Response Timeout</b>	Specifies the maximum time interval, in seconds, in which a response to a command is expected from a connected slave device. If the interval is set too short, valid replies may be missed. If set too long, retries sent by the Master may be interpreted as a reply and cause valid exchanges to be missed. This will typically be set to a slightly lower value than the Modbus master timeout. (E.g. 1.5 seconds.)
-------------------------	--

Table 13. Modbus Interface

## 8. CONFIGURE THE MQTT INTERFACE

MQTT is an OASIS and IEC/ISO standard.

The configuration interface provided enables you to configure the MQTT Client that resides on the SmartSwarm 351. The MQTT Client will use this configuration to connect to, and to communicate with, an MQTT Broker. Once connected, the SmartSwarm 351 will publish data to the MQTT Broker.

The data published will be in accordance with the Rules and Topics that you will define for your Modbus environment: see chapters 9 and 10.



*The MQTT Broker is a 3rd party service: Advantech B+B SmartWorx does not provide this service. Any MQTT 3.1.1 compliant broker may be used.*



*For information on deploying MQTT in a secure manner we recommend that you refer to “MQTT and the NIST Cybersecurity Framework” which is available on the OASIS website (<http://docs.oasis-open.org/mqtt/mqtt-nist-cybersecurity>).*

Modbus  
Decoder  
MQTT

\* Required Field

MQTT

Host:

Port:

Username:

Password:

Client ID:

Timeout (secs):

Retry Interval (secs):

Keep Alive (secs):

Reliability:

Clean Session:

---

Enable TLS:  ▼

Verify Server Cert:

Mutual Authentication:

**Server Root CA Cert** +

**Client Certificate** +

**Client Private Key** +

Passphrase:

**Last Will & Testament** +

Setting	Description
Host	Enter the IP address of the MQTT broker. This is the address that the SmartSwarm 351 will publish MQTT Topics to.
Port	TCP/IP port used by the MQTT broker. The default port for MQTT is 1883 When TLS is enabled the default port is 8883.  Ensure that the port you use matches the port on the MQTT Broker.

Username/Password	<p>Username and Password fields may be used to authenticate and authorize the client when connecting.</p> <p>These fields are optional: If you setup your MQTT broker to require them, then you will require them here also.</p> <p>The password is sent in plaintext if it isn't encrypted or hashed by implementation, or if TLS is not used.</p> <p>We recommend that you use username and password together with a secure transport (i.e. Enable TLS).</p> <p>Alternatively, you may choose to use the Client Certificate method for authentication. This is best if your broker supports it. In this case, no username and password are needed.</p>
Client ID	<p>Unique identifier, used by the broker to uniquely identify each client. This field is optional, and may be left blank.</p> <p>The broker uses it for identifying the client and the current state of the client. If you don't need a state to be held by the broker, in MQTT 3.1.1 it is possible to send an empty Client ID. This results in a connection without any state. A condition is that Clean Session is true, otherwise the connection will be rejected.</p> <p>We recommend that you use a random number.</p>
Timeout	<p>Connection timeout in seconds. That is, the number of seconds that the client will persist in attempting to make an initial connection with the broker.</p>
Retry Interval	<p>The number of seconds after a QoS=1 or QoS=2 message has been sent that the publisher will wait before retrying when no response is received.</p>
Keep Alive	<p>The Keep Alive is a time interval used by the client to ensure the connection with the broker is kept open. The client sends a PING request to the broker as specified by this time interval. The broker responds with PING Response and this mechanism will allow both sides to determine if the other one is still alive and reachable.</p>
Reliability	<p>This is a Boolean value that controls how many messages can be in-flight simultaneously.</p> <p>Setting Reliable to True means that a published message must be completed (acknowledgements received) before another can be sent.</p> <p>Setting this flag to false allows up to 10 messages to be in-flight. This can increase overall throughput in some circumstances.</p>
Clean Session	<p>True: The broker won't store anything for the client and will also purge all information from a previous persistent session. This is required to be True if there is no Client ID used.</p> <p>This is the recommended setting for SmartSwarm 351.</p> <p>False: The broker will store all subscriptions for the client and also all missed messages, when subscribing with Quality of Service (QoS) 1 or 2.</p>

Enable TLS	Enable TLSv1.2 as the secure transport layer. Security settings must match the broker settings.
Client Certificate	Valid X.509 Certificate containing the client's public key. This certificate will be sent to the broker when the SSL/TLS session is established.
Client Private Key	Valid Private key corresponding to the Client Certificate. This is not exchanged with the broker or any third party: it is only used locally on the SmartSwarm 351 device.  When using a Client Certificate, this field is required.
Passphrase	Optional passphrase for the private key.
Verify Server Cert	If this box is ticked, when the SSL/TLS session is established, the SmartSwarm 351 client will attempt to verify that the broker's certificate is trusted. The Server Root CA Cert must be provided in the field below. (For test purposes, it may be useful to disable this option. But it should be enabled for secure applications).
Server Root CA Cert	The Root CA cert used to sign the broker's certificate. If Verify Server Cert is enabled, this field is required.
Mutual Authentication	Mutual authentication means that the broker will attempt to verify the client's certificate when the SSL/TLS session is established. You must ensure that the Root CA cert used to create the <i>client's</i> key-pair is on the broker.

Table 14. MQTT Interface

Last Will and Testament	
Setting	Description
Topic	Topic on which the LWT is published. If left blank the default topic is [SwarmID]/[Serial Number]/[Status]  Any subscribing clients wishing to be notified when this SmartSwarm 351 goes online and/or offline will subscribe to this topic.
Online Message	The message the broker will send to any subscribing clients when this SmartSwarm 351 comes "online" (Successfully connects to the broker).
Offline Message	The message the broker will send to any subscribing clients when this SmartSwarm 351 goes "offline" (Disconnects unexpectedly from the broker).

<p><b>QoS</b></p>	<p>The Quality of Service level is an agreement between sender and receiver of a message regarding the guarantees of delivering a message. There are 3 QoS levels in MQTT:</p> <ul style="list-style-type: none"> <li>• At most once (0)</li> <li>• At least once (1)</li> <li>• Exactly once (2)</li> </ul> <p>With QoS for MQTT, there are always two different parts of delivering a message: publishing client to broker, and broker to subscribing client.</p> <p>The QoS level for publishing client to broker depends on the QoS level the publishing client sets for the particular message.</p> <p>When the broker transfers a message to a subscribing client it uses the QoS of the subscription made by the subscribing client.</p> <p>That means that QoS guarantees can get downgraded for a particular receiving client if subscribed with a lower QoS.</p> <p>In the case of SmartSwarm 351, we only need to consider the QoS for the publishing client to broker.</p>
<p><b>Retain</b></p>	<p>This flag determines whether the message will be saved by the broker for the specified topic as the last known good value.</p> <p>New clients that subscribe to that topic will receive the last retained message on that topic instantly after subscribing.</p>

Table 15. MQTT Interface

	<p><i>These settings are the general MQTT connection settings. For each individual Rule and Topic that has an MQTT Publish associated with it, it is possible to specify the QoS, Retain, Topic and Payload for that individual Publish. This is configured within the "Rules and Topics" section..</i></p>
---	---

	<p><i>Last Will and Testament messages are sent by the broker, to subscribing clients, when any of the following cases occur.</i></p> <ul style="list-style-type: none"> <li>• <i>An I/O error or network failure is detected by the broker;</i></li> <li>• <i>The client fails to communicate within the Keep Alive time;</i></li> <li>• <i>The client closes the network connection without sending a DISCONNECT packet first;</i></li> <li>• <i>The server closes the network connection because of a protocol error</i></li> </ul>
---	--

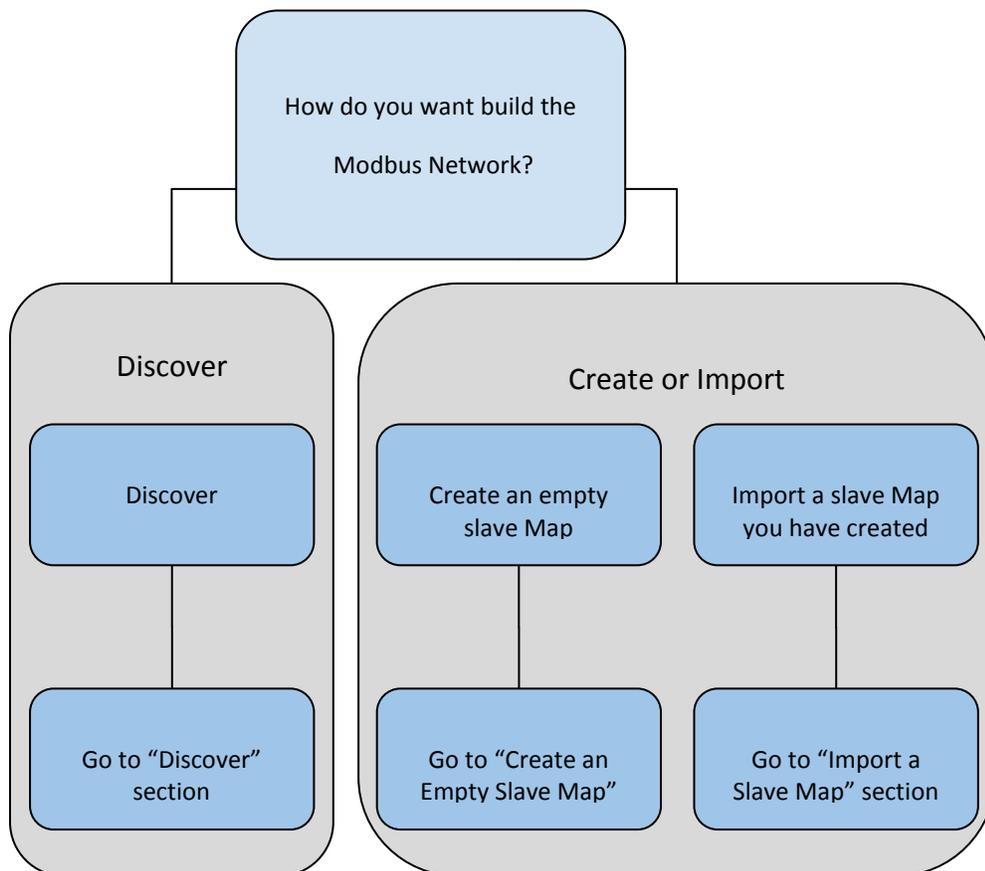
## 9. SLAVE MAPS AND ENRICHMENT

If you have some knowledge of the Modbus network that is being sniffed, you can “enrich” the Modbus data by telling the SmartSwarm device how to interpret these addresses and substitute some more meaningful strings.

There are two fundamentally different ways to enter this enrichment information:

- Discover:
  - Allow the device to automatically learn the network and the memory maps of the various slaves and present this information to you.
  - Then edit this information with any extra knowledge that you have.
- Create or Import a Slave Map:
  - Use your prior knowledge of the Modbus network and the slaves to pre-configure the device. This can be done in advance of connecting the device to a network.

It is possible to mix these two approaches in whatever way makes most sense for your application. Here, we will present one example of each.



Dashboard > Devices > Manage Device > Settings

Modbus

**Decoder**

MQTT

### Application Settings

Device ID 203-01-6500003

Application Name Modbus2MQTT

Version 0.4.9

Tag

\* Required Field

Decoder

Slaves					
Port	Slave	Name	Description	Location	

The “Decoder” interface screen displays all the Modbus Slaves currently known by SmartWorx Hub for this device.

When you first use SmartWorx Hub to enrich your new SmartSwarm device the list of Slaves will be empty.

**Slaves** Click Options icon to expand menu options

- Export Maps
- Download Templates
- Sync Maps
- New Map
- Load Map

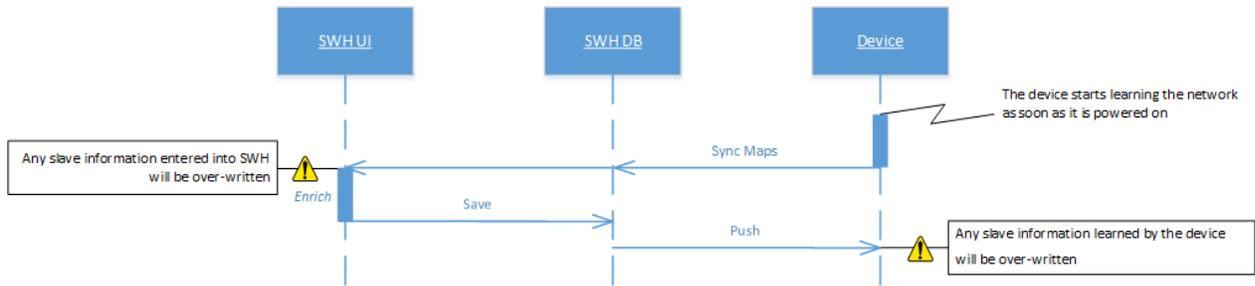
Port	Slave	Name	Description	Location
------	-------	------	-------------	----------

Option	Method	Description
<b>Export Maps</b>		Export your current maps. All of your existing Slave Maps will be exported to a .zip archive file. All maps are exported in .json format. There is one map file for each Slave. The .zip archive will be stored into your Browser's download directory.
<b>Download Templates</b>		Download a .zip archive file of map templates in both .json and Excel formats. These templates can be used to create slave map files that can then be loaded using "Load Map".
<b>Sync Maps</b>	Discover	When the SmartSwarm 351 device is connected to a live Modbus fieldbus it immediately begins to "Discover" the bus. It will automatically build up a Map of all Slaves and Registers that it sees.  You may upload this automatically discovered Modbus information from the device to SmartWorx Hub using Sync Maps. See section "Import a Slave Map".
<b>New Map</b>	Create	You may already know the slave address(es) you wish to enrich for rules-and-publish purposes.  Use this option to build only the Modbus slaves and registers that you are interested in. See section "Create an Empty Slave Map".
<b>Load Map</b>	Import	You may import your entire Modbus information using the Load Map option. You may import Maps in either .json or Excel formats. See section "Import a Slave Map".

Table 16. Slave Map options

Once you have some Slave maps in place, you may Edit or Delete ( - ) them.

9.1 DISCOVER



In Discover Mode you simply wait for your device to learn the network. The time you need to wait depends on the configuration of the Modbus Master.

After waiting for an appropriate period of time, click on “Sync Maps”. This will pull the learned information into SmartWorx Hub.

Clicking on **Sync Maps** tells the device to upload its discovered maps. This should be done after the application has discovered all slaves on the Modbus network and before enrichment data has been entered.

Any maps on the device which have already been enriched will retain their data. After syncing, the Slaves grid will be populated with the discovered maps. As no enrichment data is available, most fields will appear as undefined.

Slaves						
Port	Slave	Name	Description	Location		
RS485	1	undefined	undefined	undefined	-	Edit
RS485	2	undefined	undefined	undefined	-	Edit

Dashboard > Devices > Manage Device > Settings > Slave

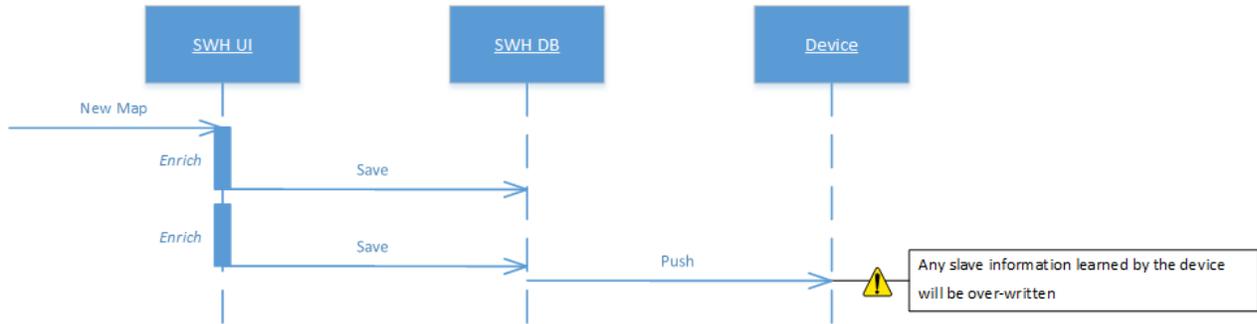
Save Push to Device Exit Editor

Meta Inputs (1x) Coils (0x) **Input Registers (3x)** Holding Registers (4x) Rules and Topics

Address	Bit Offset	Name	Alias	Data Type	Width	Zero Value	Max	Min	Scaling	Units	Num Value	Str Value	
38		undefined	undefined	UINT16	16					undefined			- +
39		undefined	undefined	UINT16	16					undefined			- +
3		undefined	undefined	UINT16	16					undefined			- +
7		undefined	undefined	UINT16	16					undefined			- +

You will now be able to edit these maps to add enrichment data and events. See Chapters 9.4, 10 on how to do this.

### 9.2 CREATE AN EMPTY SLAVE MAP



Clicking on **New Map** allows you to enter an empty slave map. Enter a slave number between 1 and 247 and click **OK**. The map is then displayed on the **Decoder** screen and may now be edited.

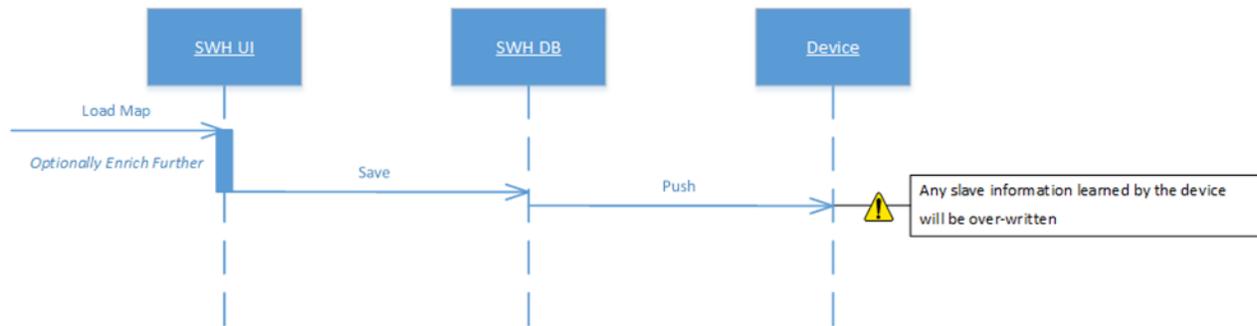
Enter a Slave between 1 and 247

**Existing slave with the same ID will be overwritten!**

OK Cancel

 Entering a Slave number which already exists on the system will overwrite the existing map

9.3 IMPORT A SLAVE MAP



You may import a complete set of pre-prepared Slave and Register information, complete with full enrichment.

If you wish to do this we recommend that you begin by exporting as much of the mapping information as you can from your Modbus control system.

You will need to manipulate your exported data into one of the formats required by the SmartWorx Hub import utility.

Click on **Load Map** to import existing slave maps. You can import multiple maps using the Ctrl and Shift keys when selecting maps to be imported. Supported formats are .json, .xls and .xlsx. Please refer to Appendix 2 for more information.

Once imported, you can edit the data in-line on SmartWorx Hub, as described in section “Editing Slaves”.

9.4 EDITING SLAVES

While editing Slaves, SmartWorx Hub will be in Editor Mode.

9.4.1 UNDERSTANDING YOUR SLAVE EDITOR

Dashboard > Devices > Manage Device > Settings > Slave



Editor Button	Description
<b>Save</b>	Save all current changes to the SmartWorx Hub Database. Use this button when you want to save your current edits, but you don't wish to push your edits to the device just yet. If you get interrupted, or SmartWorx Hub times out and logs you off, you will be able to resume editing from where you last Saved.
<b>Push to Device</b>	Push the current state of configuration for everything in the Slave Editor tabs to the SmartSwarm device. Use this button when you want to deploy your edits, so that they take effect on the SmartSwarm device.
<b>Exit Editor</b>	Exit Slave Editor mode. Note that when you exit the Editor you will be prompted to <b>Stay on page</b> if you have unsaved changes. Please remember to <b>Save</b> your changes as you make them, and to <b>Push</b> them as often as necessary

Table 17. Editing Slave Maps

	<i>When you exit the Editor you will be prompted to Stay or Leave page if you have unsaved changes. If you Choose Leave, your changes will not be saved.</i>
--	--

Context Button	Tab	Description
	Inputs (1x) Coils (0x) Input Registers (3x) Holding Registers (4x) Rules and Topics	Delete a row from the panel context.
	Inputs (1x) Coils (0x) Input Registers (3x)	Add a row to the panel context.

	Holding Registers (4x) Rules and Topics	
<b>Save Rules</b>	Rules and Topics	Save all current rules/events changes to the SmartWorx Hub database.  Use this button while you're still editing your Rules and Topics.
<b>Push Rules</b>	Rules and Topics	Push all existing (saved) rules/events to the SmartSwarm device, so that they can take effect.

Table 18. Editing Slave Maps - Rules

	<i>Save does not write anything to the SmartSwarm device. You must push your changes to the device to apply the changes.</i>
---	--

### 9.4.2 META DATA

The Meta tab allows you to add information about the Modbus slave. Some of this information is then automatically used (by default) to define the MQTT topic on which data will be published.

Dashboard > Devices > Manage Device > Settings > Slave

Save Push to Device Exit Editor

Meta Inputs (1x) Coils (0x) Input Registers (3x) Holding Registers (4x) Rules and Topics

Description	Install Date	Location	Manufacturer	Name	Product Code	Byte Order	Version
Fan	4 Apr 2016	Warehouse	B+B	Fan1	BB-XXX	Swap Words only	1.0

	<p><i>If Meta Data is populated the custom "MQTT Topic" will be composed of the text from the Location, Description and Name fields as entered in the Meta Data tab: E.g. &lt;Location&gt;/&lt;Description&gt;/&lt;Name&gt;</i></p> <p><i>When considering topic design, it is important that the topic hierarchy is carefully chosen to facilitate searching and filtering using wildcards. In the example shown in the screenshot, the "MQTT Topic" is Warehouse/Fan/Fan1.</i></p> <p><i>The custom value that the "MQTT Topic" takes by default, using Meta Data, is not the same thing as the "Default Topic".</i></p>
---	--

Field	Map Property	Description
<b>Description</b>	description	User-defined text e.g. Fan
<b>Install Date</b>	installation_date	Date the slave was installed on your network

<b>Location</b>	location	User-defined text e.g. Warehouse/Room401
<b>Manufacturer</b>	manufacturer	User-defined text (Usually the manufacturer of the slave)
<b>Name</b>	name	User-defined text e.g. AHU Fan
<b>Product Code</b>	product_code	User-defined text e.g. Wil-Flex-450
<b>Byte Order</b>	value_byte_order	This is how Floating Point and 32-bit data is ordered when it is transmitted by the slave within the register. No Swap Swap Bytes and Words Swap Bytes only Swap Words only
<b>Version</b>	version	User-defined text e.g. 4.1

Table 19. Meta Data tab



An MQTT Topic hierarchy can be specified by using the '/' character inline in the text of the meta data fields. This hierarchy is not limited to 3 levels of <Location>/<Description>/<Name>

E.g. the text "Warehouse/HoldingArea/Room401" can be entered into the Location field.

Click on a cell to enter edit-mode. Use the tab key to navigate between cells. If the data entered is invalid, the cell color will change to red. Pressing the ESC key while in edit-mode will restore the original value.

Address must be between 0 and 65535

Save Push to Device Exit Editor

Meta Inputs (1x) Coils (0x) Input Registers (3x) Holding Registers (4x) Rules and Topics

Address	Bit Offset	Name	Alias	Annotation	Data Type	Width	Zero Value	Max	Min	Scaling	Units	Num Value	Str Value	
7000d					INT16	16	0	65535	0	1				- +

Changes are automatically saved when moving to a new tab.

### 9.4.3 REGISTERS

The application supports the 4 Modicon register types i.e. Coils, Discrete Inputs, Input Registers and Holding Registers.

Register Type	Address Range	Modicon Address
<b>Coil (0x)</b>	0-65535	000001 to 065536
<b>Discrete Input (1x)</b>	0-65535	100001 to 165536
<b>Input Registers (3x)</b>	0-65535	300001 to 365536
<b>Holding Registers (4x)</b>	0-65535	400001 to 465536

Table 20. Register Types



On SmartWorx Hub, for each register type, the address entered corresponds with the Modbus register offset for that specific register type.  
E.g. Holding Register address 5 corresponds to register 400006

Click on the appropriate tab to view registers.

### 9.4.3.1 INPUT REGISTERS AND HOLDING REGISTERS

Dashboard > Devices > Manage Device > Settings > Slave

Save Push to Device Exit Editor

Meta Inputs (1x) Coils (0x) Input Registers (3x) Holding Registers (4x) Rules and Topics

Address	Bit Offset	Name	Alias	Data Type	Width	Zero Value	Max	Min	Scaling	Units	Num Value	Str Value	
4352	0	Speed		INT16	16	0	0	0	1	rpm			- +
4353	0	Type of Application		INT16	16	0	0	0	1				- +
4354	0	Measuring System		INT16	16	0	0	0	1				- +
4355	0	Interval Time		INT16	16	0	0	0	1				- +
4356	0	Position Selector, 3		INT16	16	0	0	0	1				- +
4357	0	Position Selector, 2		INT16	16	0	0	0	1				- +
4358	0	Position Selector, 1		INT16	16	0	0	0	1				- +
4359	0	Position Selector, 0		INT16	16	0	0	0	1				- +
4360	0	Filter Span Parameter		INT16	16	0	0	0	1				- +
4361	0	Filter Coefficient		INT16	16	0	0	0	1				- +
4362	0	RS485 Address		INT16	16	0	0	0	1				- +
4363	0	RS485 Baud Rate		INT16	16	0	0	0	1				- +
4364	0	ID Code of User 1		INT16	16	0	0	0	1				- +
4366	0	ID Code of User 3		INT16	16	0	0	0	1				- +

Item	Map Property	Description
Address	address	Index from the register type base address
Bit Offset	address_offset	Starting position within the register, counting from the least significant bit.  The default is 0, which is appropriate for all non-Enum Data Types.  An Enum Data Types is used to represent a register that is used for multiple purposes (e.g. using individual bits, or bit-fields).  In the case of an Enum Data Type, the Bit Offset, Width, Num Value, and String Value fields are relevant.

<b>Name</b>	name	<p>A description of the register function e.g. Energy Meter</p> <p>The Name field is not used for any algorithmic purpose within the Device: it will become part of the enrichment-data published for this Register.</p>
<b>Alias</b>	alias	<p>An alternate name for the register e.g. Power Usage</p> <p>The Alias field is not used for any algorithmic purpose within the Device: it will become part of the enrichment-data published for this Register.</p>
<b>Data Type</b>	datatype	<p>Dropdown list of data types. See section "Data Types".</p>
<b>Width</b>	length	<p>Register data width. For 16 and 32-bit data types this cannot be changed. For other types the width can be from 1 to 32.</p> <p>For Enum types this is used in conjunction with Bit Offset. Width specifies the bit-width (e.g. "4" specifies a bit-width of 4 bits) within the register; Bit Offset specifies the starting position of those 4 bits within the register.</p>
<b>Zero Value</b>	zero_value	<p>Zero calibration value for this register.</p> <p>This is an important value in converting from the Modbus Register Value to a context-aware Enriched Value.</p> <p>The equation used to enrich the Modbus data is:</p> $\text{Enriched\_Value} = (\text{Modbus\_Register\_Value} / \text{Scaling}) + \text{Zero\_Value}$ <p>See examples given below.</p>
<b>Max</b>	max	<p>The expected Maximum Enriched value for this register.</p> <p><i>E.g. If we know that the register value represents temperature with a maximum value of 100 degrees Celsius, the value entered here would be "100".</i></p> <p>This Max field value is not used for any algorithmic purpose within the Device. It will become part of the enrichment-data published for this register. It should be used as an indicator of what the maximum enriched value is expected to be.</p> <p><b>Exception to this rule:</b> For <b>Counter</b> Data Type, Max is the rollover value of the register.</p>

<p><b>Min</b></p>	<p>min</p>	<p>The expected Minimum Enriched value for this register.</p> <p><i>For example, if we know that the register value represents temperature with a minimum value of 0 degrees Celsius, the value entered here would be "0".</i></p> <p>This Min field value is not used for any algorithmic purpose within the Device. It will become part of the enrichment-data published for this register.</p> <p>It should be used as an indicator of what the minimum enriched value is expected to be.</p>
<p><b>Scaling</b></p>	<p>scaling</p>	<p>Data scaling factor to be used</p> <p>This is an important value in converting from the Modbus Register Value to a context-aware Enriched Value.</p> <p>The equation used to enrich the Modbus data is:</p> $\text{Enriched\_Value} = (\text{Modbus\_Register\_Value} / \text{Scaling}) + \text{Zero\_Value}$ <p>See the examples given below.</p>
<p><b>Units</b></p>	<p>units</p>	<p>The unit of data e.g. kWh, Hz, Deg. C, Deg. F</p> <p><i>For example, if we know that the register value represents temperature, and we want to represent the temperature in the Celsius scale, we would enter "Deg C" here.</i></p> <p>This Units field value is not used for any algorithmic purpose within the Device. It will become part of the enrichment-data published for this register.</p>
<p><b>Num Value</b></p>	<p>num</p>	<p>This field is only relevant for Enum data types.</p> <p>The Num Value field enables us to specify a numeric value that we can use to add contextualized meaning to each relevant state of bit-field data.</p> <p><i>E.g. A 16 bit register may be represented by an Enum Data Type. Each bit of the register might have a unique and significant meaning. For each bit there can be two possible states: 0 or 1. We can create a row in the Register Table that represents each bit in the Register (using Bit Offset and Width fields). For each row, we can apply meaning:          Num Value = 0; Str Value = "Valve Closed"          Num Value = 1; Str Value = "Valve Open"</i></p> <p>This Num Value field is not used for any algorithmic purpose within the Device. It will become part of the enrichment-data published for this register.</p>

<p><b>Str Value</b></p>	<p>val</p>	<p>This field is only relevant for Enum data types.</p> <p>The Str Value field enables us to specify a string value that we can use to add contextualized meaning to each relevant state of bit-field data.</p> <p><i>E.g. A 16 bit register may be represented by an Enum Data Type. Each bit of the register might have a unique and significant meaning. For each bit, there can be two possible states: 0 or 1. We can create a row in the Register Table that represents each bit in the Register (using Bit Offset and Width fields). For each row, we can apply meaning:                      Num Value = 0; Str Value = "Valve Closed"                      Num Value = 1; Str Value = "Valve Open"</i></p> <p>This Str Value field is not used for any algorithmic purpose within the Device. It will become part of the enrichment-data published for this register.</p>
-------------------------	------------	--

Table 21. Input Register and Holding Register editable fields

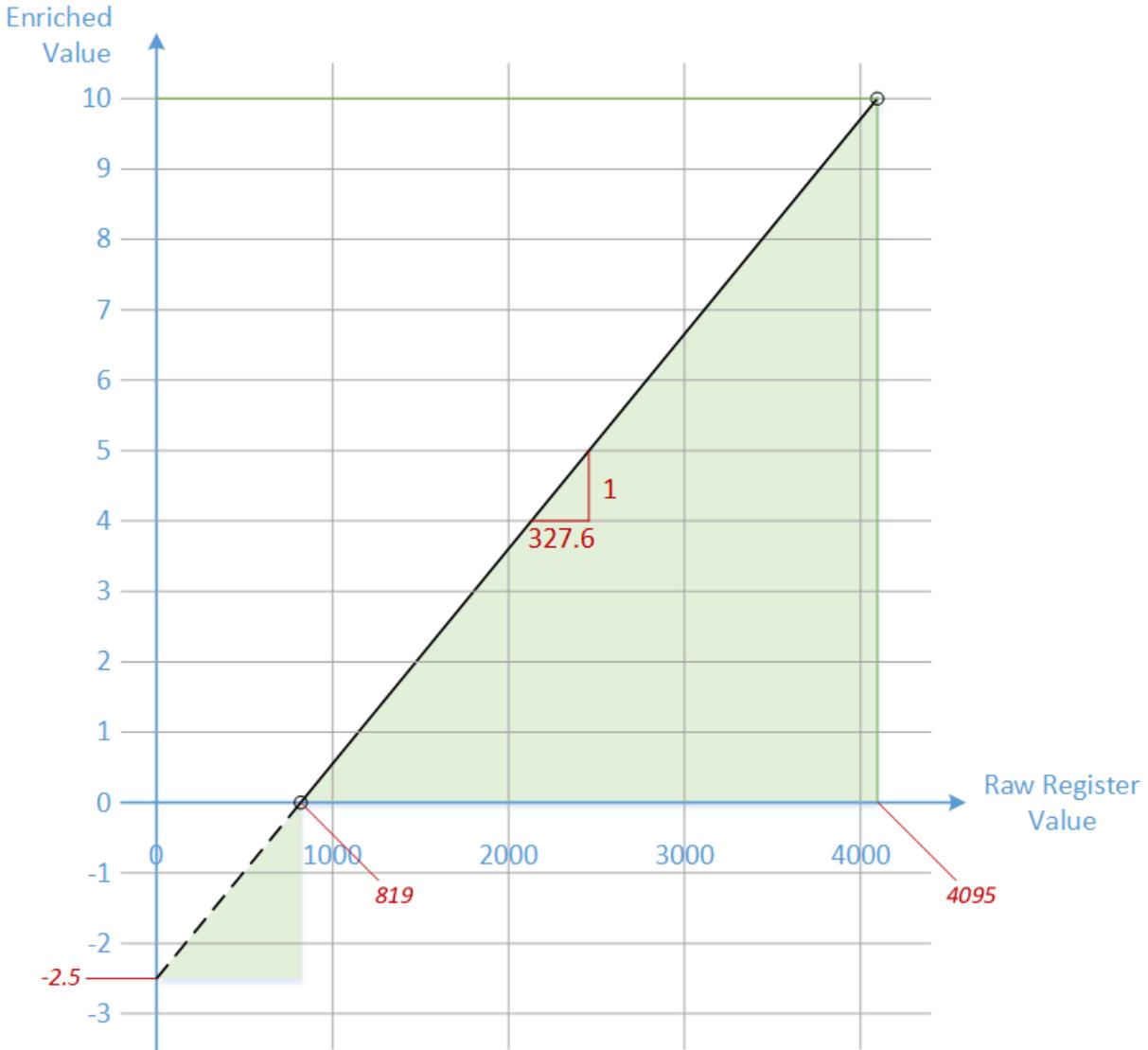
9.4.3.1.1 EXAMPLE USE OF ZERO VALUE AND SCALING

Consider an example where a Modbus slave uses a 12-bit ADC to digitize a 4-20mA loop sensor which is measuring temperature. The register value ranges from 819 to 4095, corresponding to a temperature range of 0 to +10°C.

The enriched value will be calculated as

$$(\text{Modbus Register Value} / \text{Scaling}) + \text{Zero Value}$$

It is recommended to draw the graph of input (raw Modbus register value) and output (enriched value) in order to calculate the Scaling and Zero Value. Note that we assume that the relationship is linear.



From the graph, we can calculate the slope as

$$\text{Slope} = \frac{10 - 0}{4095 - 819} = 0.00305$$

The Scaling factor is the inverse of this. It can also be thought of as the "One in X" gradient of the graph.

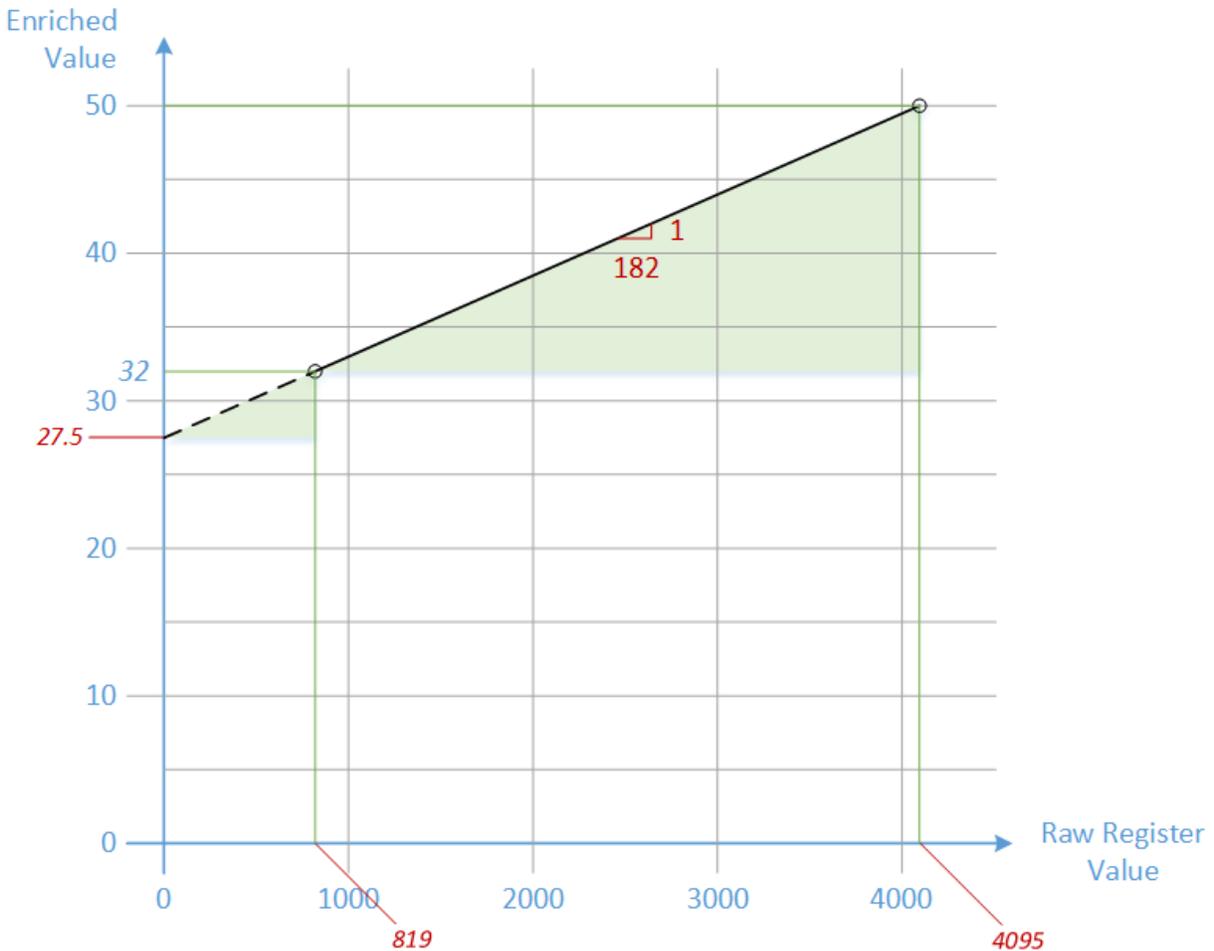
$$\text{Scaling} = \frac{1}{0.00305} = 327.6$$

The Zero Value is where the graph intersects the Y-axis. It can also be thought of as the Enriched value that corresponds to a Raw Modbus Register value of 0.

$$\text{Slope} = \frac{0 - \text{Zero Value}}{819 - 0}$$

$$\text{Zero Value} = -\text{Slope} \times 819 = -2.5$$

We can repeat this same example using an Enriched Value in degrees Fahrenheit instead of degrees Celsius. The register value ranges from 819 to 4095, corresponding to a temperature range of 32 to 50F.



Now the calculations are as follows:

$$\text{Slope} = \frac{50 - 32}{4095 - 819} = 0.00549$$

The Scaling factor is the inverse of this. It can also be thought of as the “One in X” gradient of the graph.

$$\text{Scaling} = \frac{1}{0.00549} = 182$$

The Zero Value is where the graph intersects the Y-axis. It can also be thought of as the Enriched value that corresponds to a Raw value of 0.

$$Slope = \frac{32 - Zero\ Value}{819 - 0}$$

$$Zero\ Value = 32 - Slope \times 819 = +27.5$$

Here's what these two examples would look like on SmartWorx Hub:

Address	Bit Offset	Name	Alias	Data Type	Width	Zero Value	Max	Min	Scaling	Units	Num Value	Str Value	
5	0	Temperature in Celsius	Celsius Example	UINT16	16	-2.5	10	0	327.6	Deg C			- +
6	0	Temperature in Fahrenheit	Fahrenheit Example	UINT16	16	27.5	50	32	182	Deg F			- +

### 9.4.3.2 DISCRETE INPUTS AND COILS

Dashboard > Devices > Manage Device > Settings > Slave

Save Push to Device Exit Editor

Address	Name	Alias	Str 0 Value	Str 1 Value	
40	Door 1	Room104	Closed	Open	- +
41	Door 2	Room104	Closed	Open	- +
42	Switch 1	SW43	Off	On	- +

Item	Map Property	Description
Address	address	Index from register type base address
Name	name	A description of the register function e.g. Pump Status
Alias	alias	An alternate name for the register e.g. Heating Pump 1
Str 0 Value	val0	Enriched string for a value of zero e.g. OFF
Str 1 Value	val1	Enriched string for a value of one e.g. ON

Table 22. Discrete Input and Coil editable fields

### 9.4.4 DATA TYPES

Data Type	Address	Bit Offset	Name	Alias	Width	Zero Value	Max	Min	Scaling	Units	Num Value	Str Value
ENUM	✓	✓	✓	✓	✓						✓	✓
UINT16	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		

INT16	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
UINT32	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
INT32	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
FLOAT32	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
STRING	✓	✓	✓	✓	✓	✓						
COUNTER	✓	✓	✓	✓	✓	✓	✓		✓	✓		

Table 23. Data Types and Field Values

Registers which are discovered will have no meaningful information filled in.

Some cells such as Num Value and Str Value are only applicable to ENUM data types and will have no effect if filled in for other data types. Moving the mouse over a cell will indicate if the cell is valid for the selected data type.

#### 9.4.4.1 ENUM

Combination of bits representing a range of distinct named values. See section “Editing Registers”.

#### 9.4.4.2 UNT16

Unsigned 16 bit integer, range 0 - 65535 .

#### 9.4.4.3 INT16

Signed 16 bit integer, range -32,768 to 32,767 .

#### 9.4.4.4 UINT32

Unsigned 32 bit integer, range 0 to 4,294,967,295 .

#### 9.4.4.5 INT32

Signed 32 bit integer, range -2,147,483,648 to 2,147,483,647 .

#### 9.4.4.6 FLOAT32

Signed 32 bit floating point, range -3.4E+38 to +3.4E+38 .

#### 9.4.4.7 STRING

Collection of Registers representing 8-bit ASCII characters. The width of this field should be set to the length of the string multiplied by 8.

#### 9.4.4.8 COUNTER

Combination of bits representing a roll-over counter. For example an 8-bit counter would range from 0 to 255.

When it reaches 255, the next increment will cause it to roll-over to 0 again. A 16-bit counter would range from 0 to 65535.

For rules purposes, the counter is assumed to always increment, never decrement. For example, for an 8-bit counter a reading of 254 followed by a reading of 4 would be treated as an increment of 6: 254 - 255 - 0 - 1 - 2 - 3 - 4...

The max value is not forced to be the maximum value determined by the bit width of the counter. It can be less if required. For example, if an 8-bit counter has a max value of 100, then the internal counter is assumed to rollover at 100. This allows digital counters to be handled, such as an electricity meter or vehicle odometer.

The nature of a counter also has implications for the type of event that can be triggered by it. See section “Events (WHEN)”.

### 9.4.5 ADDING REGISTERS

To add a register click the + icon. This will create a new row with the same values as the current row. Enter the Address and fill in the other fields as required.

### 9.4.6 EDITING REGISTERS

#### 9.4.6.1 DISCRETE INPUTS AND COILS

Discrete Inputs and Coils can only have a value of zero or one. These can be enriched using the Str 0 Value and Str 1 Value columns.

Dashboard > Devices > Manage Device > Settings > Slave

The screenshot shows a web interface for configuring a device. At the top, there are three buttons: "Save", "Push to Device", and "Exit Editor". Below these are several tabs: "Meta", "Inputs (1x)", "Coils (0x)", "Input Registers (3x)", "Holding Registers (4x)", and "Rules and Topics". The "Inputs (1x)" tab is selected, displaying a table with the following data:

Address	Name	Alias	Str 0 Value	Str 1 Value	
40	Input-1	SW-43	OFF	ON	- +
41	Input-2	Door 1	Open	Closed	- +
42	Input-3	Tank 1	Empty	Full	- +

#### 9.4.6.2 INPUT REGISTERS AND HOLDING REGISTERS

By default, discovered Holding and Input registers will have a DataType of UINT16 when displayed. The user should select the correct Data Type for each register.

##### 9.4.6.2.1 THE ENUM DATA TYPE

“ENUM” stands for enumerated type. It means that the register can have a finite set of *named* values. When entering ENUM data types, an entry must be made for each enum value. i.e. If there are 2 enums representing ON and OFF, then 2 rows must be entered for the register.

In the example below, Register 614 is an ENUM type which uses Bit 0 to encode 2 possible states: 0 and 1, which have enriched values of “OFF” and “ON”. Note how we use the combination of “Bit offset” and “Width” to specify bit zero of the register.

Dashboard > Devices > Manage Device > Settings > Slave

Save Push to Device Exit Editor

Meta Inputs (1x) Coils (0x) Input Registers (3x) Holding Registers (4x) Rules and Topics

Address	Bit Offset	Name	Alias	Data Type	Width	Zero Value	Max	Min	Scaling	Units	Num Value	Str Value	
614	0	Drive Enable		ENUM	1						0	OFF	- +
614	0	Drive Enable		ENUM	1						1	ON	- +

In the next example, Register 40 is also an ENUM type which uses the first 3 bits of the register.

- Bit 0 has 2 states: "Pump Off" and "Pump On".
- Bit 1 has 2 states: "Normal Operation" and "Min Speed"
- Bit 2 has 2 states: "Normal Operation" and "Max Speed".

Note how one register configured as a "bit field" in this way can hold several independent states at the same time.

Dashboard > Devices > Manage Device > Settings > Slave

Save Push to Device Exit Editor

Meta Inputs (1x) Coils (0x) Input Registers (3x) Holding Registers (4x) Rules and Topics

Address	Bit Offset	Name	Alias	Data Type	Width	Zero Value	Max	Min	Scaling	Units	Num Value	Str Value	
1	0	Set Value		UINT16	16	0	200	0	1	0.5%			- +
40	0	Pump Command		ENUM	1						0	Pump Off	- +
40	0	Pump Command		ENUM	1						1	Pump On	- +
40	1	Pump Command		ENUM	1						0	Normal Operation	- +
40	1	Pump Command		ENUM	1						1	Min Speed	- +
40	2	Pump Command		ENUM	1						0	Normal Operation	- +
40	2	Pump Command		ENUM	1						1	Max Speed	- +
42	0	Operation Mode		ENUM	16	0	140	0	1		0	Unknown	- +
42	0	Operation Mode		ENUM	16	0	140	0	1		1	Fixed speed	- +
42	0	Operation Mode		ENUM	16	0	140	0	1		2	Reserved	- +
42	0	Operation Mode		ENUM	16	0	140	0	1		3	dp-c regulation	- +
42	0	Operation Mode		ENUM	16	0	140	0	1		4	dp-v regulation	- +
42	0	Operation Mode		ENUM	16	0	140	0	1		5	Reserved	- +
42	0	Operation Mode		ENUM	16	0	140	0	1		6	dp-T regulation	- +
42	0	Operation Mode		ENUM	16	0	140	0	1		140	PID Control	- +

Another common ENUM scenario uses the value of the whole register to encode a single state. In the above screenshot Register 42 represents one state variable that can have multiple possible values.



*The Min, Max, Units, Num Value and Str Value fields are not used for any algorithmic purpose within the Device. The data entered for these fields will become part of the enriched-data published for the register.*

*The Zero Value and Scaling fields are used to scale the raw Modbus value into an enriched value: For*

*ENUM data types this could alter the intended meaning of Modbus register. Unless you're sure, we recommend leaving these fields in their default states (Zero Value = 0; Scaling = 1).*

Dashboard > Devices > Manage Device > Settings > Slave

Save Push to Device Exit Editor

Meta	Inputs (1x)	Coils (0x)	Input Registers (3x)	Holding Registers (4x)	Rules and Topics								
Address	Bit Offset	Name	Alias	Data Type	Width	Zero Value	Max	Min	Scaling	Units	Num Value	Str Value	
65	0	Flow Valve		ENUM	2						1	Valve Open	- +
65	0	Flow Valve		ENUM	2						0	Valve Closed	- +
65	0	Flow Valve		ENUM	2						2	Valve In Transition	- +
65	2	Pressure Valve		ENUM	2						1	Valve Open	- +
65	2	Pressure Valve		ENUM	2						0	Valve Closed	- +
65	2	Pressure Valve		ENUM	2						2	Valve In Transition	- +
14	8	Control Room Door		ENUM	2						0	Door Closed	- +
14	8	Control Room Door		ENUM	2						1	Door Open	- +
14	8	Control Room Door		ENUM	2						2	Door Opening	- +
14	8	Control Room Door		ENUM	2						3	Door Closing	- +

In the above screenshot Registers 65 and 14 are ENUM types.

Register 65 uses two 2-bit values to represent the state of two Valves.

The 2-bit width allows up to four states for the Valves: in the example shown, only three states are defined.

Register 14 uses a 2-bit width to encode 4 states to represent the current status of a control-room door.

- Register 65, Bit Offset 0, has three states: "Valve Open", "Valve Closed", "Valve in Transition"
- Register 65, Bit Offset 2, has three states: "Valve Open", "Valve Closed", "Valve in Transition"
- Register 14, Bit Offset 8, has four states: "Door Closed", "Door Open", "Door Opening", "Door Closing".

### 9.4.7 DELETING REGISTERS

To delete a register, Click the  icon. A confirmation dialog is displayed. Click **OK** to delete the register. The register is now removed.

 No changes will be made to the device until the **Push to Device** button has been clicked

10. RULES AND TOPICS

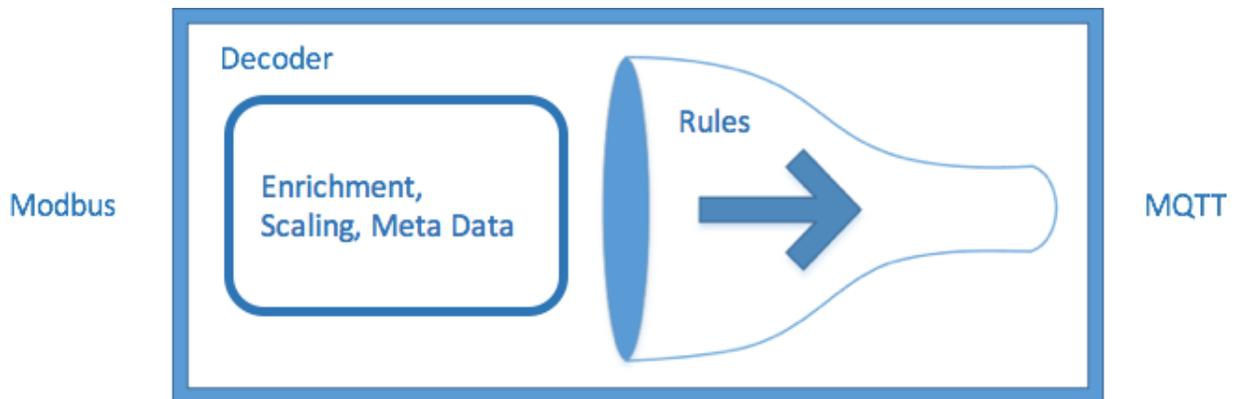
10.1 INTRODUCTION

The SmartSwarm 351 converts Modbus data to MQTT.

In the process it adds data enrichment, including scaling factors and meta-data.

By default, all data is blocked and nothing is published on MQTT until you specifically allow it.

Bear in mind that even a slow serial network, running continuously, can create a lot of data. A 9600-baud network running at 50% bus utilization generates 1.5GB of raw data every month: and this is significantly increased by the enrichment process. If you are transporting the MQTT data over cellular you probably cannot afford to publish everything, and it is unlikely that your cellular connection and cloud service would keep up with the sustained, enriched-data rate.



The “Decoder” interface on SmartWorx Hub enables you to first apply enrichment for your Modbus data, and then to apply rules for your enriched data.

If you want some specific data to be published on MQTT, you must add a filter “rule”. A rule has two parts:

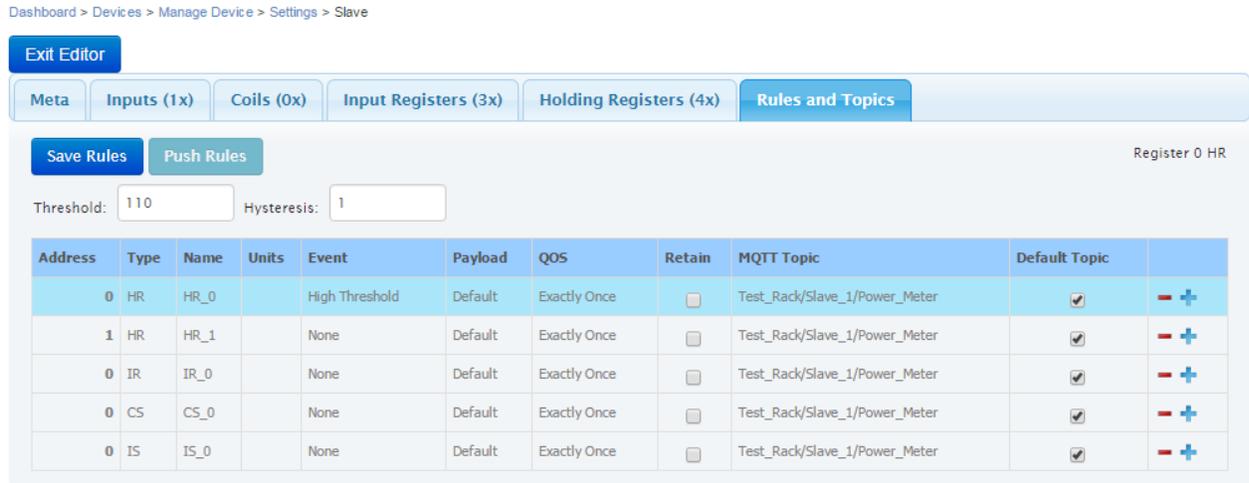
- An *event*, which determines WHEN data will be published;
- A *payload*, which determines WHAT data will be published.

	<p><i>Enrichment is done before Rules are applied. This means that all Rules created will apply to the enriched data.</i></p> <p><i>If scaling has been applied for a register during the enrichment process, the Rules you create will apply to the enriched and scaled data for that register.</i></p> <p><i>If you have not applied enrichment for a register, the Rules you create for that register will apply to the raw Modbus register data.</i></p>
--	--

The published Payload data will have all of the enrichment data included.

You then specify HOW you want the data to be published. For example, on what MQTT topic, with what QOS, etc.

The Rules and Topics screen is visible as a Tab in the editor for each slave. For example:



By default, only one event is displayed for each available register.

To add another event, click the '+' sign on the required register. This will add a duplicate row for that register. Edit the event details and click **Save Rules**.

When saving rules, an error message will be displayed if any required parameters for an event are missing.

Once rules have been saved, click **Push Rules** to apply the rules to the device.

The following fields are editable in the Rules and Topics tab:

Item	Description
Event	Select the event type from a dropdown menu
Payload	Select what to publish from a dropdown menu
QOS	Quality of Service
Retain	Instructs the broker to retain the last published message on this topic
MQTT Topic	Topic to publish on
Default Topic	The default topic will always be Swarm_ID/Device_ID/Port_ID/Slave_ID/Register_Type/Event E.g. 0/700000/1/1/HR/SCHEDULING. This can be turned on or off
-	Delete this rule
+	Add a new rule to this register

Table 24. Rules and Topics fields

10.2 EVENTS (WHEN)

On any row in the Rules and Topics table, click the **Event** field to see the drop-down list of available events:

Dashboard > Devices > Manage Device > Settings > Slave

Exit Editor

Meta Inputs (1x) Coils (0x) Input Registers (3x) Holding Registers (4x) **Rules and Topics**

Save Rules Push Rules Register 0 HR

Address	Type	Name	Units	Event	Payload	QOS	Retain	MQTT Topic	Default Topic	
0	HR	HR_0		None	ault	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
1	HR	HR_1		None	t	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
0	IR	IR_0		Read	t	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
0	CS	CS_0		Change	t	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
0	IS	IS_0		Delta	t	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
				High Threshold	t	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
				Low Threshold	t	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
				High Rate	t	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
				Low Rate	t	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
				Scheduled	t	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
				Global Read	t	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
				Global Change	t	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +

**Note:** Only one instance of each event type should be added to a register. Adding multiple events of the same type will have no effect, as only the first instance of the event will take effect.

The following table summarizes the available event types:

Event	Description
None	The default selection. Nothing will be published.
Read	Trigger when the register is observed on the bus. Use with extreme caution as this can result in a substantial amount of MQTT data being published.
Change	Trigger when the enriched register value changes from the last observed value.
High Threshold	Trigger when the enriched register value goes above a threshold.
Low Threshold	Trigger when the enriched register value goes below a threshold.
Delta	Trigger when the enriched register value changes by a specified amount from the last published value.
High Rate	Trigger when the rate-of-change of the enriched register value increases by more than a specified amount in a specified period.
Low Rate	Trigger when the rate-of-change of the enriched register value decreases by more than a specified amount in a specified period.
Scheduled	Trigger periodically, on a user-defined interval.
Global Read	As for "Read", but applies to ANY register of this type.
Global Change	As for "Change", but applies to ANY register of this type.

Table 25. Event Types

The following table summarizes how each event type may be used with each register type:

Event	Inputs, Coils	Input Registers, Holding Registers			
		ENUM	Numeric <sup>1</sup>	STRING	COUNTER
Read	✓	✓	✓	✓	✓
Change	✓	✓	✓	✓	✓
High Threshold			✓		
Low Threshold			✓		
Delta			✓		✓
High Rate			✓		✓
Low Rate			✓		✓
Scheduled	✓	✓	✓	✓	✓
Global Read	✓		✓	✓	✓
Global Change	✓		✓	✓	✓

Table 26. Events and Data Types: cross-reference

<sup>1</sup> “Numeric” means any of the following data types: UINT16, INT16, UINT32, INT32, FLOAT32

Each event type is discussed in more detail in the following sections.

### 10.2.1 READ

Trigger a publish when a register has been read (or written) by the Modbus master.

This event is applicable to any register type and any data type: For a counter it returns the raw value of the register, not the accumulated value.

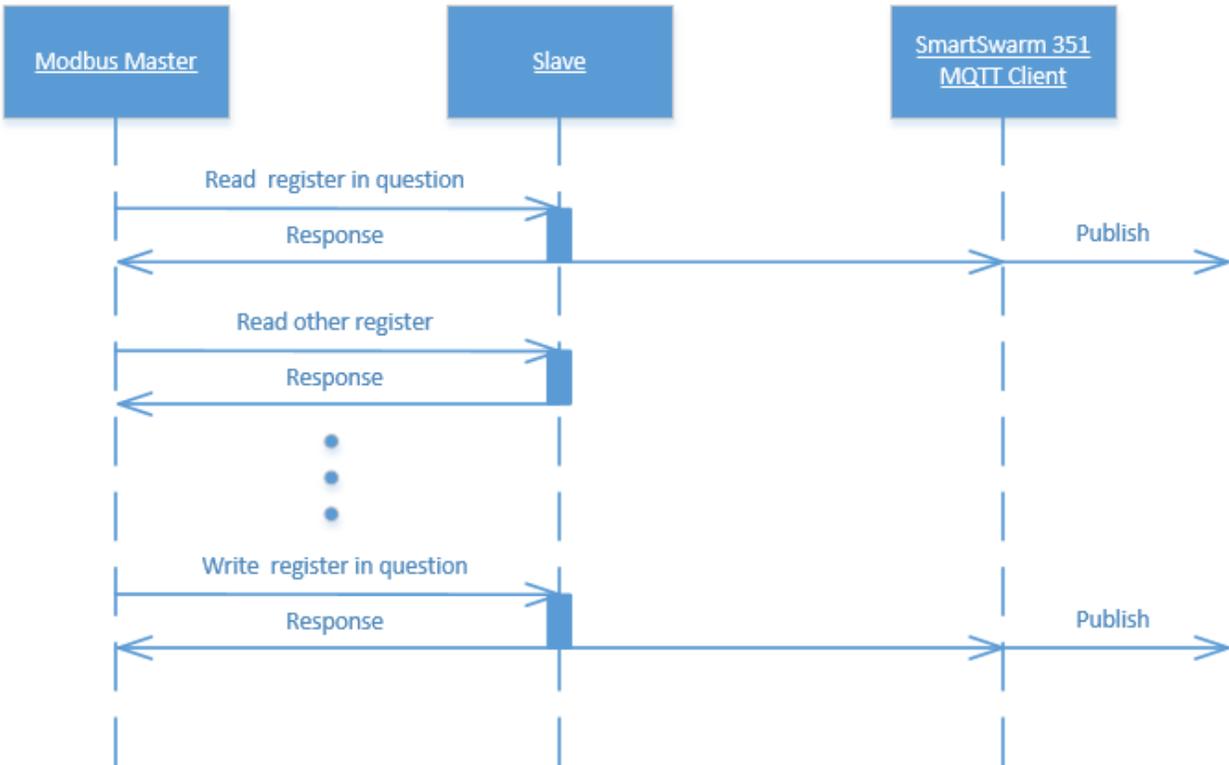
Event	Inputs, Coils	Input Registers, Holding Registers			
		ENUM	Numeric	STRING	COUNTER
Read	✓	✓	✓	✓	✓

Table 27. Read Event

<sup>1</sup> “Numeric” means any of the following data types: UINT16, INT16, UINT32, INT32, FLOAT32

In SmartWorx Hub, when you select this Event type. No further configuration is required:

Address	Type	Name	Units	Event	Payload	QOS	Retain	MQTT Topic	Default Topic	
0	HR	HR_0		Read	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +



### 10.2.2 CHANGE

Trigger a publish when an enriched register value changes.

This event is applicable to any register type and any data type:

	Inputs, Coils	Input Registers, Holding Registers			
		ENUM	Numeric	STRING	COUNTER
Change	✓	✓	✓	✓	✓

Table 28. Change Event

In SmartWorx Hub, when you select this Event type on an Input or a Coil, no further configuration is required. However, if it is selected on an Input Register or a Holding Register, an additional field appears at the top of the table:

Dashboard > Devices > Manage Device > Settings > Slave

Exit Editor

Meta Inputs (1x) Coils (0x) Input Registers (3x) Holding Registers (4x) Rules and Topics

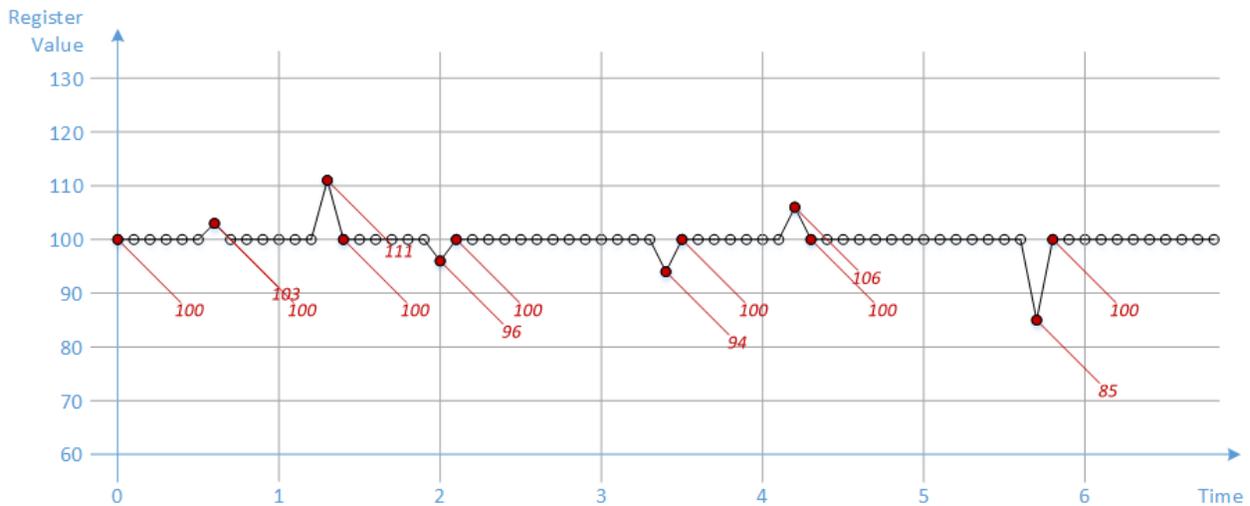
Save Rules Push Rules Register 0 HR

Change by: % percent

Address	Type	Name	Units	Event	Payload	QOS	Retain	MQTT Topic	Default Topic	
0	HR	HR_0		Change	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
1	HR	HR_1		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
0	IR	IR_0		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
0	CS	CS_0		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
0	IS	IS_0		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +

In the “Change by” field, you may enter a percentage value. The MQTT publish will only be triggered if the enriched register value has changed by **more** than this value. The default value is 0%, which means that ANY change will constitute a trigger event.

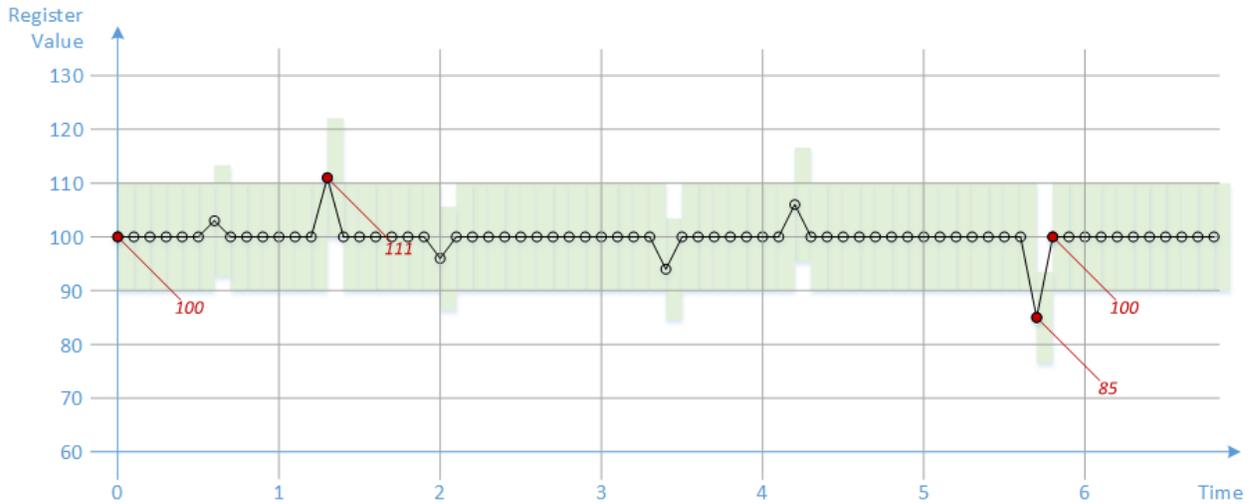
The following example shows a register that usually has a numeric value of 100, but with occasional deviations. The red annotations show when MQTT publishes will be triggered, assuming that a Change rule is applied, with “Change by” = 0%:



Note that we get only 13 MQTT messages, as opposed to 68 if we had used a Read rule.

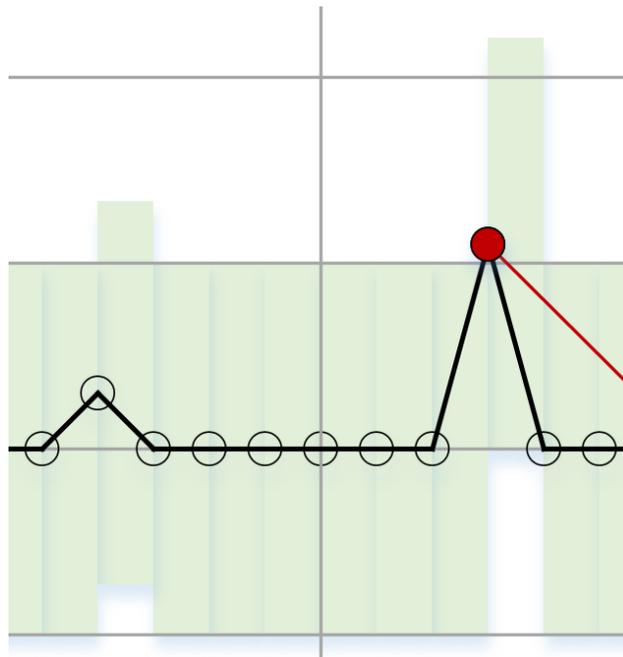
For analog data it probably does not make sense to create a Change rule with “Change by” = 0%, as process noise will inevitably cause the least-significant bits to change all the time. This will trigger a lot of MQTT publishes, which may lead to a large cellular bill. The “Change by” field can be used to reject noise.

Continuing the example above, if we apply a Change rule with “Change by” = 10% we can reduce the number of MQTT messages from 13 to 4:



It is important to remember that the change is measured with respect to the last time the register was *observed on Modbus*, not to the last time it was *published on MQTT*.

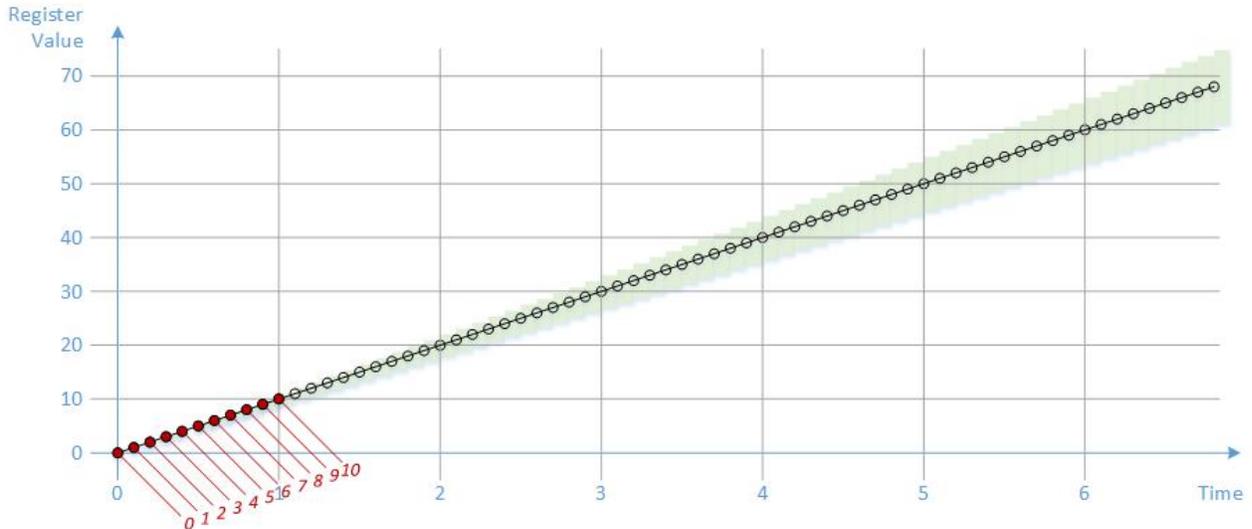
In the diagram above, the green bars represent the ±10% change that is dynamically calculated after each new sample is received.



An MQTT publish will be triggered only if the next sample value lies outside of the allowed range, represented by the green bar.

For the “Change” event type, only a percentage or *relative* change can be specified. That means the actual change in value required to trigger an event will vary.

The following example shows a numeric register that ramps monotonically from 0, incrementing by 1 on each Modbus master access. The red annotations show when MQTT publishes will be triggered, assuming that a Change rule is applied, with "Change by" = 10%.



- If the register has not been seen on the bus before, then the first value of 0 is considered to be a change of 100%. So the value of 0 triggers a publish.
- If the current enriched register value is 0, then any non-zero value is considered to be a change of 100%. So the value of 1 triggers a publish.
- The next change from 1 to 2 represents a 100% change, which is greater than 10%.
- A change from 2 to 3 represents a 50% change, and so on.
- A change from 10 to 11 represents a 10% change, which is not greater than 10%. The value of 11 does not trigger a publish.
- As the enriched register value increases further, the absolute change of 1 becomes smaller in percentage terms. No further MQTT publishes are triggered.

For a slave waveform like this you probably want to specify an *absolute* change. See the Delta event type below.

### 10.2.3 DELTA

Trigger when the enriched register value changes from the last *published* value.

This event is only applicable to Input Registers and Holding Registers, not Discrete Inputs and Coils.

It is only applicable to numeric data types, and the counter data type:

It is only applicable to numeric data types, and the counter data type:

	Inputs, Coils	Input Registers, Holding Registers			
		ENUM	Numeric	STRING	COUNTER

Delta			✓		✓
-------	--	--	---	--	---

Table 29. Delta Event

**Note:** As explained in the Counter section, a counter is always increasing. A drop in value is assumed to indicate a single roll-over event. Therefore, for an 8-bit counter (range 0 - 255), a change in value from 254 to 253 is an increase of 255, not a decrease of 1.

Like the Change event, you can enter a “Change by” value after selecting the Delta event type on a register:

Dashboard > Devices > Manage Device > Settings > Slave

Exit Editor

Meta | Inputs (1x) | Coils (0x) | Input Registers (3x) | Holding Registers (4x) | Rules and Topics

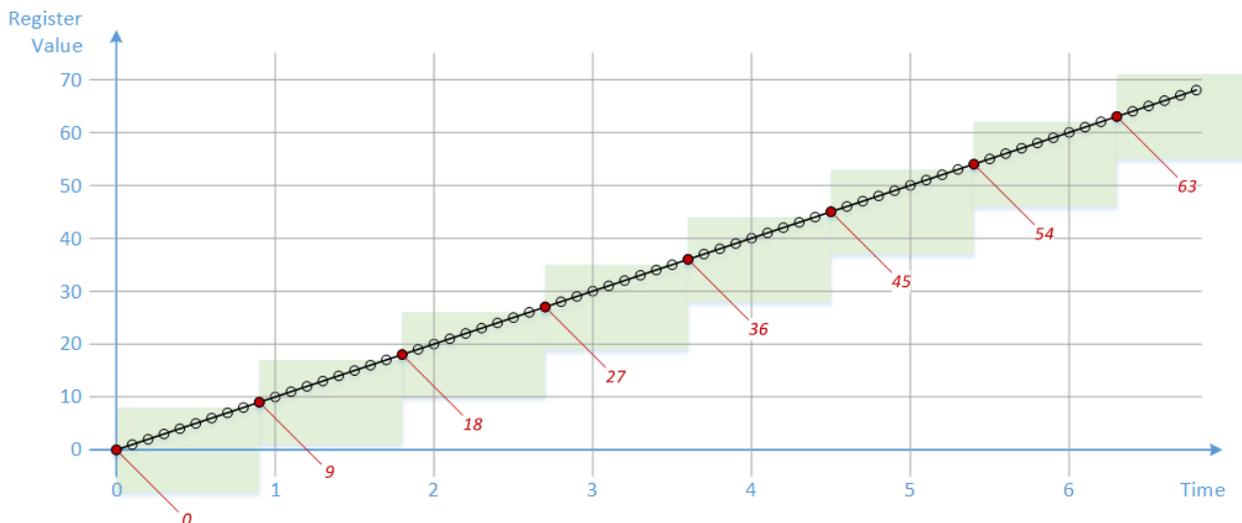
Save Rules | Push Rules Register 0 HR

Change by:

Address	Type	Name	Units	Event	Payload	QOS	Retain	MQTT Topic	Default Topic	
0	HR	HR_0		Delta	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
1	HR	HR_1		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
0	IR	IR_0		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
0	CS	CS_0		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
0	IS	IS_0		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +

The MQTT publish will only be triggered if the enriched register value has changed by **greater than or equal to** this value, *compared to the last time it was published*.

The following example shows a register that ramps monotonically from 0, incrementing by 1 on each Modbus master access. The red annotations show when MQTT publishes will be triggered, assuming that a Delta rule is applied, with “Change by” = 9.



10.2.4 HIGH THRESHOLD

Trigger when the enriched register value increases above a fixed threshold.

This event is only applicable to Input Registers and Holding Registers, not Discrete Inputs and Coils.

It is applicable to numeric data types only. It is not applicable to a counter. A counter is a rollover data type and so we have no record of its true accumulated value on which to base the threshold.

	Inputs, Coils	Input Registers, Holding Registers			
		ENUM	Numeric	STRING	COUNTER
High Threshold			✓		

Table 30. High Threshold Event

When you select the High Threshold event type on a register, you must enter two fields:

Dashboard > Devices > Manage Device > Settings > Slave

[Exit Editor](#)

Meta | **Inputs (1x)** | Coils (0x) | Input Registers (3x) | Holding Registers (4x) | Rules and Topics

Save Rules | Push Rules Register 0 HR

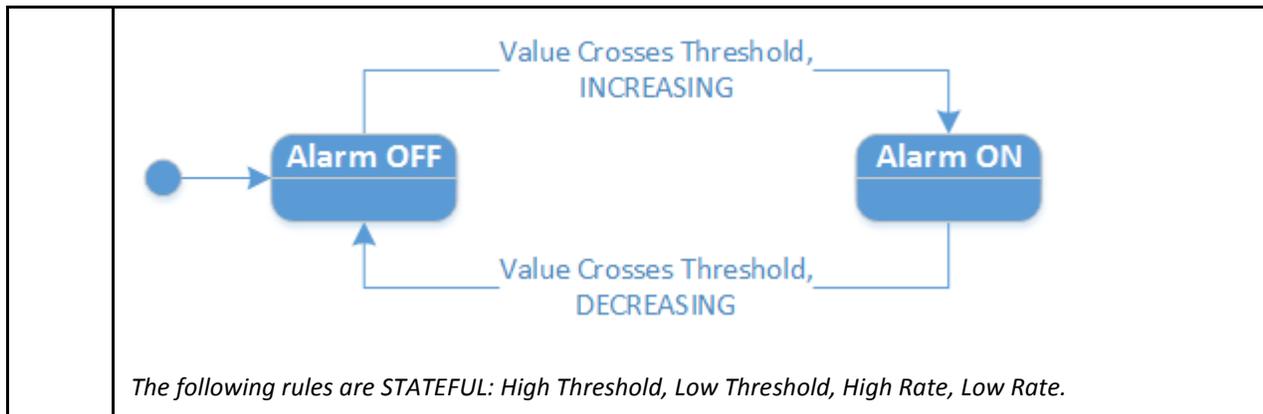
Threshold:  Hysteresis:

Address	Type	Name	Units	Event	Payload	QOS	Retain	MQTT Topic	Default Topic	
0	HR	HR_0		High Threshold	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
1	HR	HR_1		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
0	IR	IR_0		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
0	CS	CS_0		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
0	IS	IS_0		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +

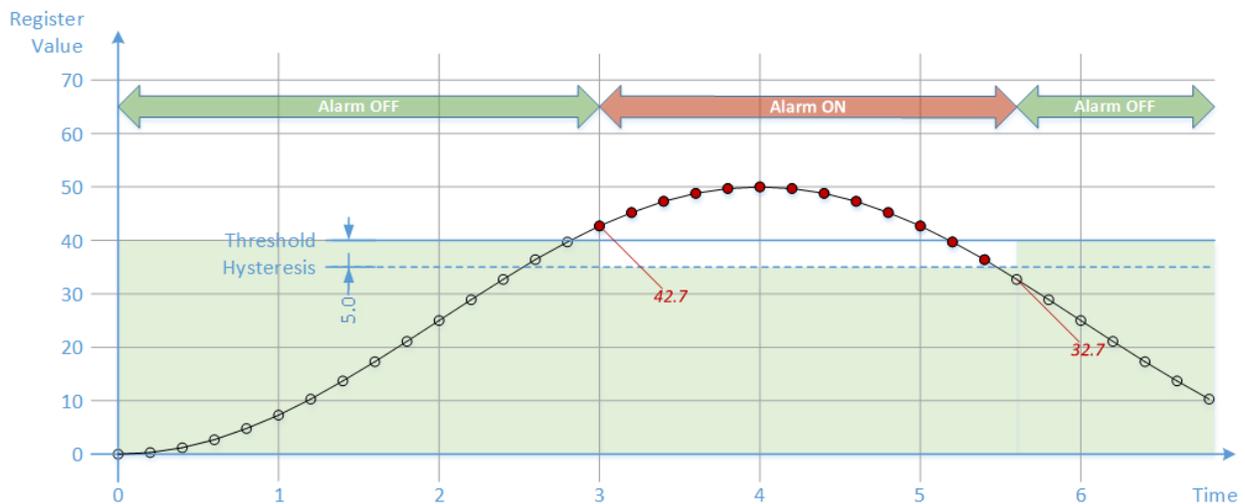
- The **Threshold** field specifies the value above which the register must increase in order to trigger an MQTT publish.
- The **Hysteresis** field allows you to prevent multiple MQTT publishes due to process noise. Once the register has crossed the Threshold value in the positive (increasing) direction, it must cross the value [Threshold - Hysteresis] in the negative direction before it is considered to have re-crossed.



**NOTE:** This rule is significantly different from the previous rules discussed above, because it is STATEFUL. For example, when the High Threshold is crossed in either direction an MQTT publish will be triggered. But if the register remains above (or below) the threshold no further messages will be published. An MQTT publish will occur only on state transitions.



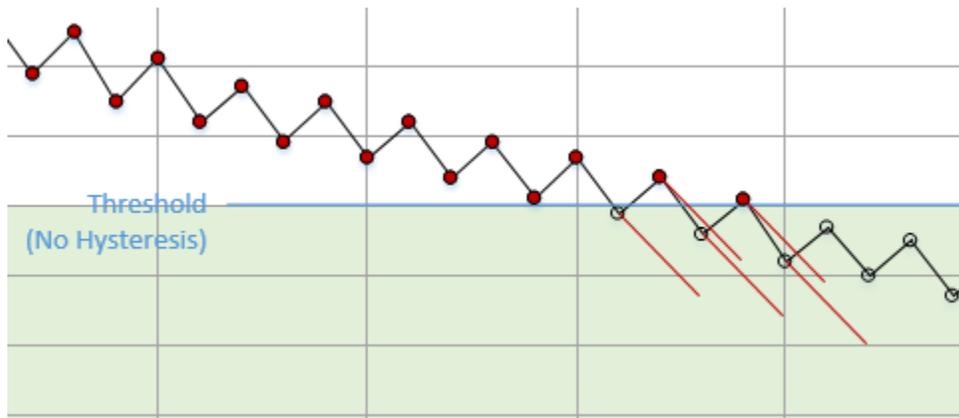
The following example shows a register that follows a sinusoidal waveform. The Modbus Master is polling the register every 200ms. Assume that a “High Threshold” rule is applied, with Threshold = 40 and Hysteresis = 5. The red dots show samples which are above the Threshold. The red callouts show when MQTT publishes will be triggered:



Note how only two MQTT messages are published:

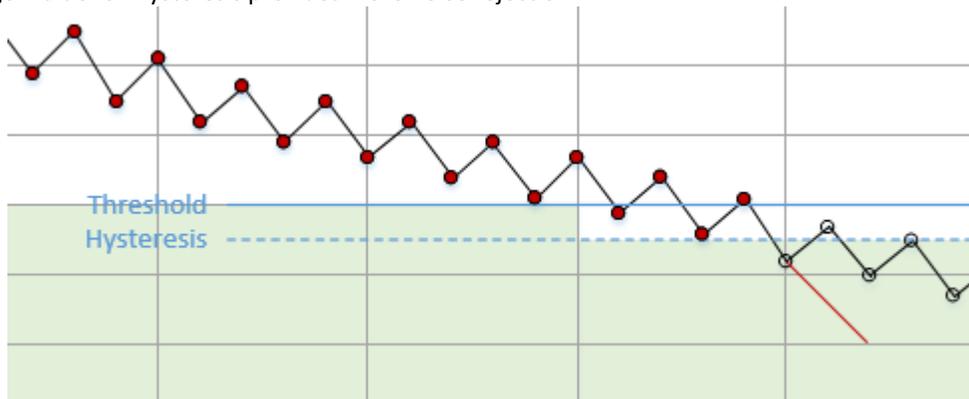
- One when the value changes from below the threshold to above;
- One when the value changes from above the threshold to below (minus the hysteresis).

The default value of the Hysteresis field is 0 (zero), which means no hysteresis:



Note how 5 MQTT publishes are caused by the multiple threshold crossings.

Using a bigger value for Hysteresis provides more noise rejection:



### 10.2.5 LOW THRESHOLD

Trigger when the enriched register value decreases below a fixed threshold..

This event is only applicable to Input Registers and Holding Registers, not Discrete Inputs and Coils.

It is applicable to numeric data types only. It is not applicable to a counter. A counter is a rollover data type and so we have no record of its true accumulated value on which to base the threshold.

	Inputs, Coils	Input Registers, Holding Registers			
		ENUM	Numeric	STRING	COUNTER
Low Threshold			✓		

Table 31. Low Threshold Event

When you select the Low Threshold event type on a register, you must enter two fields:

Dashboard > Devices > Manage Device > Settings > Slave

Exit Editor

Meta Inputs (1x) Coils (0x) Input Registers (3x) Holding Registers (4x) Rules and Topics

Save Rules Push Rules Register 0 HR

Threshold: 15 Hysteresis: 5

Address	Type	Name	Units	Event	Payload	QOS	Retain	MQTT Topic	Default Topic	
0	HR	HR_0		Low Threshold	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
1	HR	HR_1		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
0	IR	IR_0		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
0	CS	CS_0		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
0	IS	IS_0		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +

- The **Threshold** field specifies the value below which the register must decrease in order to trigger an MQTT publish.
- The **Hysteresis** field allows you to prevent multiple MQTT publishes due to process noise. Once the register has crossed the Threshold value in the negative (decreasing) direction it must cross the value [Threshold + Hysteresis] in the positive direction before it is considered to have re-crossed.

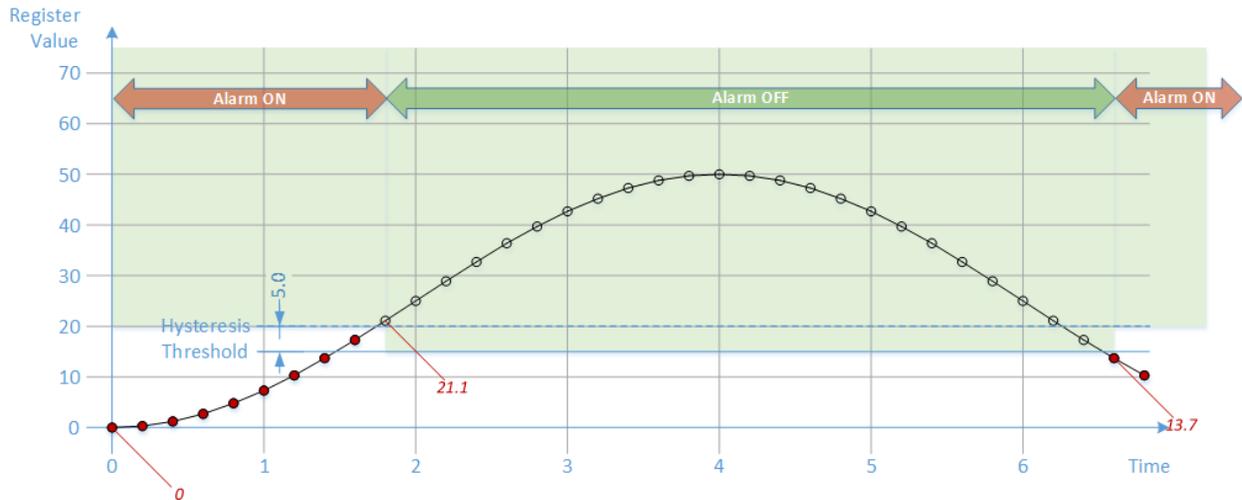
*NOTE: This rule is STATEFUL. For example, when the Low Threshold is crossed, in either direction, an MQTT publish will be triggered. But if the register remains below (or above) the threshold, no further messages will be published. An MQTT publish will occur only on state transmissions:*

```

    graph LR
      Start(( )) --> AlarmOFF[Alarm OFF]
      AlarmOFF -- "Value Crosses Threshold, DECREASING" --> AlarmON[Alarm ON]
      AlarmON -- "Value Crosses Threshold, INCREASING" --> AlarmOFF
  
```

*The following rules are STATEFUL: High Threshold, Low Threshold, High Rate, Low Rate.*

The following example shows the same register as before. The red callouts show when MQTT publishes will be triggered, assuming that a “Low Threshold” rule is applied, with Threshold = 15 and Hysteresis = 5:



Notes:

- On start-up, if the first value is already below the threshold, that is considered a state transition. In this case, the first value of 0 will trigger an MQTT publish.
- When the value increases above the threshold (plus the hysteresis), we publish again.
- When the value decreases below the threshold, we publish again.

10.2.6 HIGH RATE

The “High Rate” rule will trigger on “high rate of change”.

Trigger when the enriched register value has changed at a rate *greater than* a certain rate.

Note: A counter is always increasing. A drop in value is taken as a rollover. Therefore, for an 8 bit counter (range 0 - 255) a change in value from 254 to 253 is an increase of 255 not a decrease of 1

This event is only applicable to Input Registers and Holding Registers, not Discrete Inputs and Coils.

It is only applicable to numeric data types and the counter data type:

**Note:** a counter is always increasing. A drop in value is taken as a rollover. Therefore, for an 8 bit counter (range 0 - 255), a change in value from 254 to 253 is an increase of 255 and not a decrease of 1.

	Inputs, Coils	Input Registers, Holding Registers			
		ENUM	Numeric	STRING	COUNTER
High Rate			✓		

Table 32. High Rate Event

When you select the “High Rate” event type on a register, you must enter a “Change” value:

Dashboard > Devices > Manage Device > Settings > Slave

Exit Editor

Meta Inputs (1x) Coils (0x) Input Registers (3x) Holding Registers (4x) Rules and Topics

Save Rules Push Rules Register 0 HR

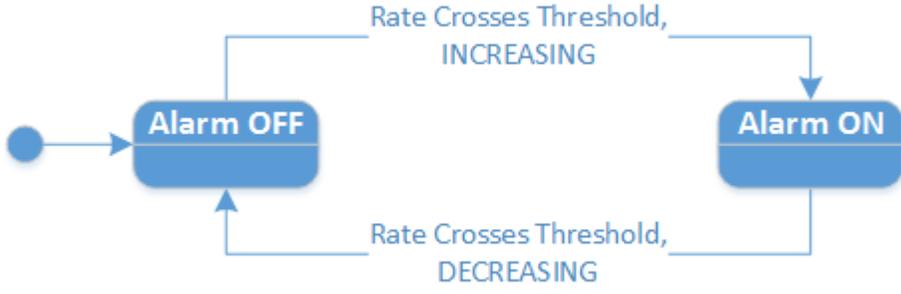
Change: 10 per second

Address	Type	Name	Units	Event	Payload	QOS	Retain	MQTT Topic	Default Topic	
0	HR	HR_0		High Rate	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
1	HR	HR_1		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
0	IR	IR_0		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
0	CS	CS_0		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
0	IS	IS_0		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +

The Change value represents the instantaneous rate of change per second. The MQTT publish will only be triggered if the enriched register value has changed *faster* than this.



**NOTE:** This rule is STATEFUL. For example, when the Rate Threshold is crossed in either direction an MQTT publish will be triggered. But if the rate of change remains below (or above) the threshold no further messages will be published. An MQTT publish will only occur on state transitions:

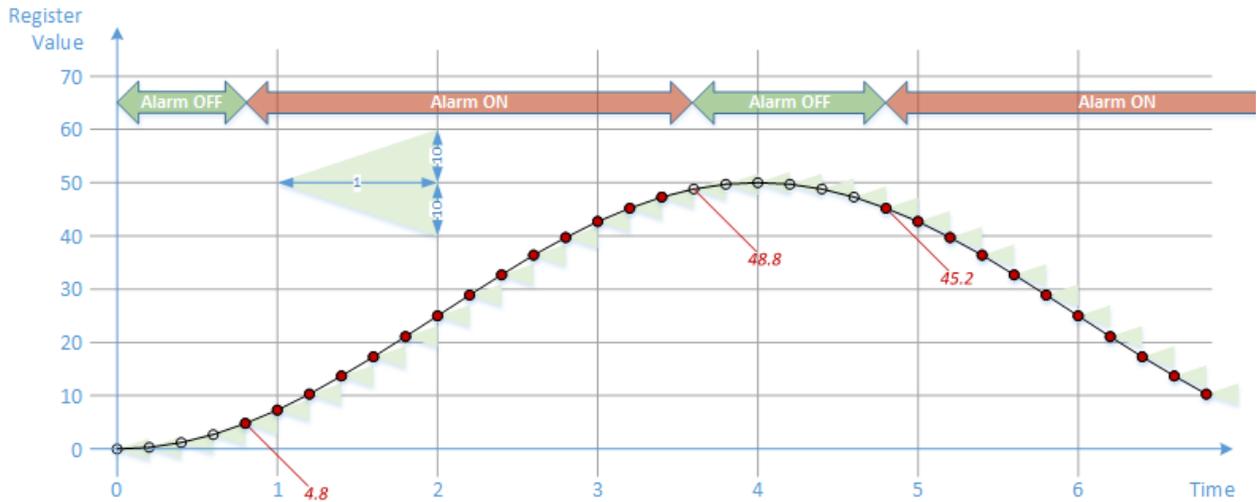


```

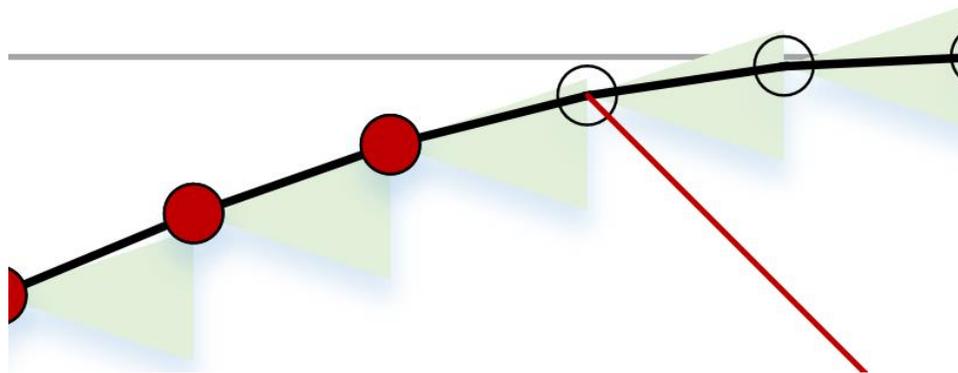
graph LR
    Start(( )) --> AlarmOff[Alarm OFF]
    AlarmOff -- "Rate Crosses Threshold, INCREASING" --> AlarmOn[Alarm ON]
    AlarmOn -- "Rate Crosses Threshold, DECREASING" --> AlarmOff
    
```

The following rules are STATEFUL: High Threshold, Low Threshold, High Rate, Low Rate.

The following example shows a register that follows a sinusoidal waveform. The Modbus Master is polling the register every 200ms. Assume that a “High Rate” rule is applied, with Change = 10 per second. The red dots show samples that are above the Rate Threshold. The red callouts show when MQTT publishes will be triggered:

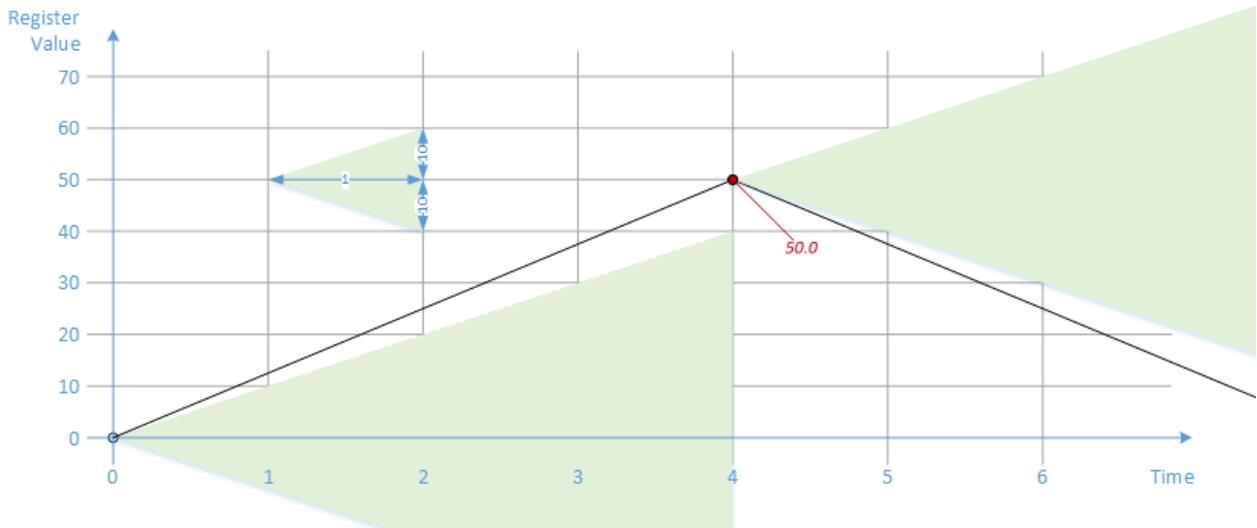


In the diagram above the green triangles represent the “10 per second” rate of change against which each new sample is compared when it is received. At the top and bottom of a sine-wave the rate of change is low, so the trigger condition is not met.



Even though the Rate limit is specified in units per second, the system does not have to wait 1 second before deciding whether or not to publish. Whenever two consecutive samples are received, be they separated in time by less than or more than 1 second, the instantaneous rate of change is calculated. If the actual rate of change is greater than the programmed threshold, the MQTT publish will be triggered.

In the following example, the Modbus Master is polling the same register at a much slower rate: once every 4 seconds:



The second sample (actually, the first pair of samples) will trigger an MQTT publish, because  $\frac{50}{4} > \frac{10}{1}$

After this the system will remain in the “Alarm ON” state forever, because the rate of change is always greater than the threshold. No further MQTT messages will be published.

The “High Rate” event type is roughly equivalent to the “High Threshold” event type, but it operates on the *derivative* of the waveform. However, note that there is no equivalent for Rate “Hysteresis”. This means that if the rate of change of a register happens to coincide with the selected Rate Threshold, timing jitter and/or process noise may lead to multiple MQTT publishes due to repeated state transitions.

### 10.2.7 LOW RATE

Trigger when the enriched register value has changed at a rate *less than* a certain rate.

Note: a counter is always increasing. A drop in value is taken as a rollover. Therefore, for an 8 bit counter (range 0 - 255) a change in value from 254 to 253 is an increase of 255 and not a decrease of 1.

This event is only applicable to Input Registers and Holding Registers, not Discrete Inputs and Coils.

It is only applicable to numeric data types, and the counter data type:

**Note:** a counter is always increasing. A drop in value is taken as a rollover. Therefore, for an 8 bit counter (range 0 - 255) a change in value from 254 to 253 is an increase of 255 and not a decrease of 1.

	Inputs, Coils	Input Registers, Holding Registers			
		ENUM	Numeric	STRING	COUNTER
Low Rate			✓		

Table 33. Low Rate Event

As for “High Rate”, when you select the “Low Rate” event type on a register you must enter a “Change” value, which represents the rate of change per second:

Dashboard > Devices > Manage Device > Settings > Slave

Exit Editor

Meta Inputs (1x) Coils (0x) Input Registers (3x) Holding Registers (4x) Rules and Topics

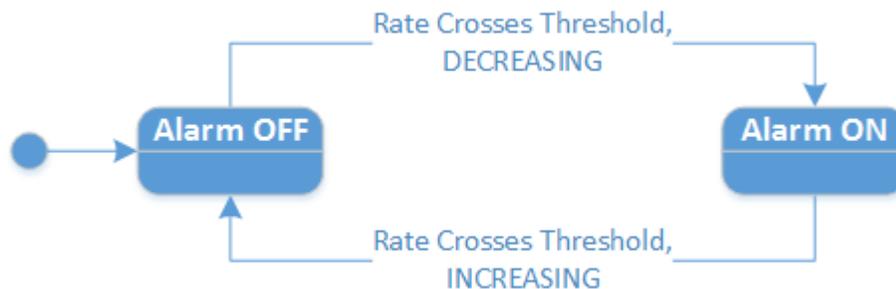
Save Rules Push Rules Register 0 HR

Change: 10 per second

Address	Type	Name	Units	Event	Payload	QOS	Retain	MQTT Topic	Default Topic	
0	HR	HR_0		Low Rate	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
1	HR	HR_1		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
0	IR	IR_0		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
0	CS	CS_0		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
0	IS	IS_0		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +

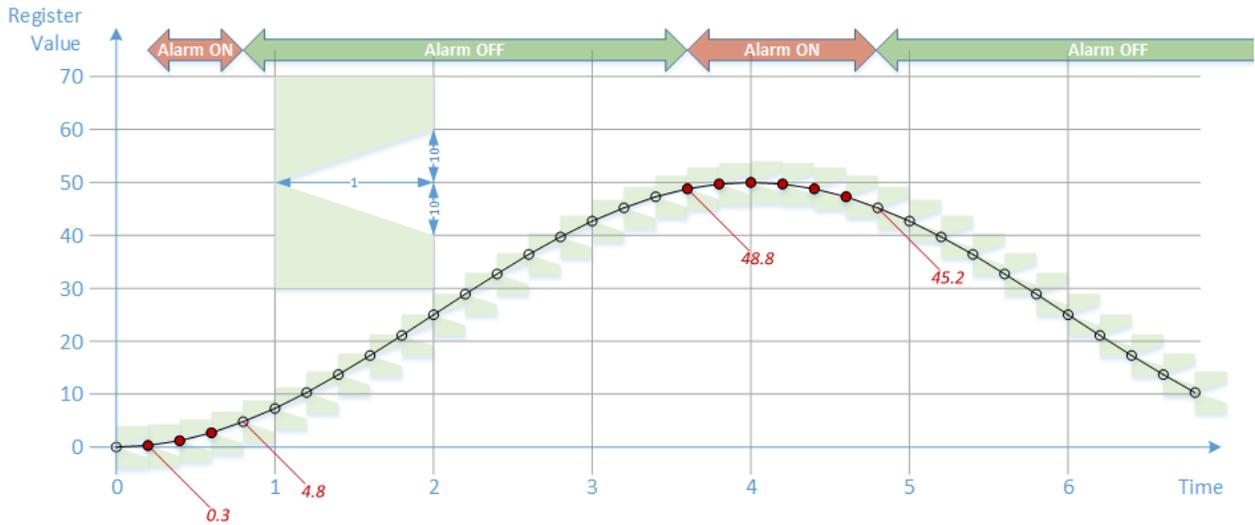


**NOTE:** This rule is STATEFUL. i.e. When the Rate Threshold is crossed in either direction, an MQTT publish will be triggered. But if the rate of change remains below (or above) the threshold, no further messages will be published. An MQTT publish will only occur on state transitions:



The following rules are STATEFUL: High Threshold, Low Threshold, High Rate, Low Rate.

The following example shows the same register as before, but with a “Low Rate” rule applied, with Change = 10 per second:



As expected, we get the complement of the previous result for the “High Rate” case.

The “Low Rate” event type is roughly equivalent to the “Low Threshold” event type, but it operates on the *derivative* of the waveform. However, note that there is no equivalent for Rate “Hysteresis”. This means that if the rate of change of a register happens to coincide with the selected Rate Threshold, timing jitter and/or process noise may lead to multiple MQTT publishes due to repeated state transitions.

### 10.2.8 SCHEDULED

This event type allows you to specify an interval at which the data will be published on MQTT, and the time within that interval when the publish will take place, e.g. every hour and 17 minutes past the hour:

Dashboard > Devices > Manage Device > Settings > Slave

Exit Editor

Meta Inputs (1x) Coils (0x) Input Registers (3x) Holding Registers (4x) Rules and Topics

Save Rules Push Rules Register 0 HR

Every minute

Address	Name	Units	Event	Payload	QOS	Retain	MQTT Topic	Default Topic	
0	CS_0		Scheduled	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
1	IS_1		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
0	CS_0		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
0	CS_0		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
0	IS_0		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +

10.2.9 GLOBAL READ

Trigger a publish when **any register** of the selected **Type** has been read (or written) by the Modbus master.

This event is applicable to any register type and any data type. For a counter it returns the raw value of the register, not the accumulated value.

	Inputs, Coils	Input Registers, Holding Registers			
		ENUM	Numeric	STRING	COUNTER
Read	✓	✓	✓	✓	✓

Table 34. Global Read Event

Otherwise, this trigger will operate as per the description in the “Read” section.

Address	Type	Name	Units	Event	Payload	QOS	Retain	MQTT Topic	Default Topic	
0	HR	HR_0		Global Read	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	

10.2.10 GLOBAL CHANGE

Trigger a publish when the value of **any register** of the selected **Type** changes.

This event is applicable to any register type and any data type: For a counter it returns the raw value of the register, not the accumulated value.

	Inputs, Coils	Input Registers, Holding Registers			
		ENUM	Numeric	STRING	COUNTER
Change	✓	✓	✓	✓	✓

Table 35. Global Change Event

Note that when a global filter is enabled, it is not possible to disable that filter for individual elements.

	<p><i>Global filters have the potential to generate a large number of MQTT publishes and should be avoided unless you are sure that this is what you intend.</i></p>
--	--

10.3 PAYLOADS (WHAT)

For every event type, you can decide WHAT to publish when that event happens.

Payload	Description
Default	Only the register that triggered the event will be published.
Slave	All registers on this slave will be published.
HR	All Holding registers on this slave will be published.
IR	All Input registers on this slave will be published.
IS	All Discrete Input registers on this slave will be published.
CS	All Coils on this slave will be published.
Range	Registers within a range will be published.

Table 36. Payload options

The “Default” payload can sometimes contain more than one register. In the case of a Global Read or Global Change rule, a single Modbus transaction may read or write more than one register. Every register that meets the Read or Change criteria will be published.

Not all payload selections are available for all event types, as the following table explains.

Event	Default	Slave	HR	IR	IS	CS	Range
Read	✓						
Change	✓						
Delta	✓	✓	✓	✓	✓	✓	✓
High Threshold	✓	✓	✓	✓	✓	✓	✓
Low Threshold	✓	✓	✓	✓	✓	✓	✓
High Rate	✓	✓	✓	✓	✓	✓	✓
Low Rate	✓	✓	✓	✓	✓	✓	✓
Scheduled	✓	✓	✓	✓	✓	✓	✓
Global Read	✓						
Global Change	✓						

Table 37. Event / Payload matrix



A payload will only be published if there has been actual data observed on the Modbus network for the defined payload type. In other words, the device will not publish “enrichment” only, without an observed “num\_value” (see payload examples below).

In the case of the “default” payload there will always be a publish to correspond with the event trigger.

10.3.1 PAYLOAD EXAMPLES

10.3.1.1 THE DEFAULT PAYLOAD

Threshold:  Hysteresis:

Address	Type	Name	Units	Event	Payload	QOS	Retain	MQTT Topic	Default Topic	
1	HR	HR_1	Deg C	High Threshold	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	<input type="text"/> <input type="text"/>

With this rule, if the enriched value of Register 1 goes above 100 the payload for register 1 will be published.

The published payload data will be in .json format.

For this HR example, the actual enriched register value within the data payload will be in `model.state.HR.0.num_value` (see the published JSON schema below).

A sample of payload published from this rule might look like this:

```
{
  "model": {
    "state": {
      "HR": [{
        "value_from": "RESPONSE",
        "published_on": "HI\\ON",
        "name": "HR_1",
        "min": 0,
        "address_offset": 0,
        "max": 100,
        "zero_value": 0,
        "num_value": 101.000000,
        "scaling": 1,
        "alias": "Water Temperature",
        "state": "VALIDATED",
        "var_pct": 16.000000,
        "at": "2016-07-14T19:29:22.089Z",
        "address": 1,
        "units": "Deg C",
        "new_value": true,
        "new_read": true
      }]
    },
    "meta": {
      "description": "Slave 1",
      "value_byte_order": "SNo",
      "name": "Power Meter",
      "installation_date": "14\\07\\2016",
      "location": "Test_Rack",
      "address": {
        "DEVID": "6500004",
        "PORTID": 1,
        "SLAVEID": 10,
        "SWMID": 0
      },
      "manufacturer": "NA"
    }
  }
}
```

```

    },
    "type": "ModbusSlave",
    "id": "10_HR_1_HI"
}

```

### 10.3.1.2 THE HR PAYLOAD

Threshold:  Hysteresis:

Address	Type	Name	Units	Event	Payload	QOS	Retain	MQTT Topic	Default Topic
1	HR	HR_1	Deg C	High Threshold	HR	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter/	<input checked="" type="checkbox"/>

With this rule, if the enriched value of Register 1 goes above 100 the payload data for all Holding registers will be published. The num\_value in the payload for each of the holding-registers will be the actual value last seen by the SmartSwarm device on the Modbus network.

If there has not been any actual data seen on Modbus for a HR register it will not appear in the payload.

The published data will be in .json format.

A sample of payload published from this rule (assuming there are 3 Holding Registers for this Slave), might look like this:

```

{
  "model": {
    "state": {
      "HR": [{
        "value_from": "RESPONSE",
        "published_on": "HI\ON",
        "name": "HR_1",
        "min": 0,
        "address_offset": 0,
        "max": 100,
        "zero_value": 0,
        "num_value": 103.000000,
        "scaling": 1,
        "alias": "Water Temperature",
        "state": "VALIDATED",
        "var_pct": 17.000000,
        "at": "2016-07-14T17:38:32.721Z",
        "address": 1,
        "units": "Deg C",
        "new_value": true,
        "new_read": true
      }, {
        "value_from": "RESPONSE",
        "published_on": "HI\ON",
        "name": "HR_2",
        "min": -10,
        "address_offset": 0,
        "max": 40,
        "zero_value": 0,
        "num_value": 61.000000,
        "scaling": 1,
        "alias": "Air Temperature",

```

```

        "state": "VALIDATED",
        "var_pct": 2.000000,
        "at": "2016-07-14T17:38:32.721Z",
        "address": 2,
        "units": "Deg C",
        "new_value": true,
        "new_read": true
    }, {
        "value_from": "RESPONSE",
        "name": "HR_3",
        "min": 0,
        "address_offset": 0,
        "max": 90,
        "zero_value": 0,
        "num_value": 0.000000,
        "scaling": 1,
        "alias": "Humidity",
        "state": "VALIDATED",
        "var_pct": 0,
        "at": "2016-07-14T17:38:32.721Z",
        "address": 3,
        "units": "%",
        "new_value": false,
        "new_read": true
    }
  ],
  "meta": {
    "description": "Slave_1",
    "value_byte_order": "SNo",
    "name": "Power_Meter",
    "installation_date": "14/07/2016",
    "location": "Test_Rack",
    "address": {
      "DEVID": "6500004",
      "PORTID": 1,
      "SLAVEID": 10,
      "SWMID": 0
    },
    "manufacturer": "NA"
  }
},
"type": "ModbusSlave",
"id": "10_HR_1_HI"
}

```

10.3.1.3 THE SLAVE PAYLOAD

Threshold:  Hysteresis:

Address	Type	Name	Units	Event	Payload	QOS	Retain	MQTT Topic	Default Topic	
1	HR	HR_1	Deg C	High Threshold	Slave	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	

With this rule, if the enriched value of Register 1 goes above 100 all registers from this Modbus slave will be published.

The num\_value in the payload for each of the registers will be the actual value last seen by the SmartSwarm device on the Modbus network.

If there are no Input Registers, Discrete Inputs, or Coils defined for the slave, then there will be place-holders in the published JSON data schema to show where the data for those register values would be, if they existed.

If there has not been any actual data seen on Modbus for a slave register it will not appear in the payload.

Here's an example payload, where there are only holding registers defined (and observed) for the Slave:

```
{
  "model": {
    "state": {
      "CS": [],
    }
  },
  "HR": [{
    "value_from": "RESPONSE",
    "published_on": "HI\ON",
    "name": "HR_1",
    "min": 0,
    "address_offset": 0,
    "max": 100,
    "zero_value": 0,
    "num_value": 113.000000,
    "scaling": 1,
    "alias": "Water Temperature",
    "state": "VALIDATED",
    "var_pct": 31.000000,
    "at": "2016-07-14T19:36:29.127Z",
    "address": 1,
    "units": "Deg C",
    "new_value": true,
    "new_read": true
  }, {
    "value_from": "RESPONSE",
    "published_on": "HI\ON",
    "name": "HR_2",
    "min": -10,
    "address_offset": 0,
    "max": 40,
    "zero_value": 0,
    "num_value": 23104.000000,
    "scaling": 1,
    "alias": "Air Temperature",
    "state": "VALIDATED",
    "var_pct": 0,
    "at": "2016-07-14T19:36:29.127Z",
    "address": 2,
    "units": "Deg C",
    "new_value": false,
    "new_read": true
  }, {
    "value_from": "RESPONSE",
    "published_on": "HI\ON",
    "name": "HR_3",
    "min": 0,
    "address_offset": 0,
    "max": 90,
    "zero_value": 0,
    "num_value": 0.000000,
    "scaling": 1,
    "alias": "Humidity",
    "state": "VALIDATED",
    "var_pct": 0,
    "at": "2016-07-14T19:36:29.127Z",
    "address": 3,
    "units": "%",
    "new_value": false,
    "new_read": true
  }
  ]
}
```

```

    }],
    "IS": [],
    "IR": []
  },
  "meta": {
    "description": "Slave_1",
    "value_byte_order": "SNo",
    "name": "Power_Meter",
    "installation_date": "14\07\2016",
    "location": "Test_Rack",
    "address": {
      "DEVID": "6500004",
      "PORTID": 1,
      "SLAVEID": 10,
      "SWMID": 0
    },
    "manufacturer": "NA"
  }
},
"type": "ModbusSlave",
"id": "10_HR_1_HI"
}

```

Here's an example payload when there are Holding Registers, Coils, Discrete Inputs and Input Registers defined (or observed) for the slave. In this example, there is 1 Coil, 3 Holding Registers, 2 Inputs, and 2 Input Registers (one of them is an ENUM, with 3 enumerated associated values):

```

{
  "model": {
    "state": {
      "CS": [{
        "published_on": "HI\OFF",
        "name": "CoilExample",
        "str_value": "OutsideBounds",
        "num_value": 0,
        "value_from": "RESPONSE",
        "alias": "This is a Coil Example",
        "state": "VALIDATED",
        "var_pct": 0,
        "new_read": false,
        "address": 1,
        "new_value": false,
        "at": "2016-07-15T15:11:00.373Z"
      }],
      "HR": [{
        "value_from": "RESPONSE",
        "published_on": "HI\OFF",
        "name": "HR_1",
        "min": 0,
        "address_offset": 0,
        "max": 100,
        "zero_value": 0,
        "num_value": 89.000000,
        "scaling": 1,
        "alias": "Water Temperature",
        "state": "VALIDATED",
        "var_pct": 17.000000,
        "at": "2016-07-15T15:16:51.407Z",
        "address": 1,
        "units": "Deg C",
        "new_value": true,
        "new_read": true
      }, {
        "value_from": "RESPONSE",
        "published_on": "HI\OFF",
        "name": "HR_2",

```

```

"value_from": "RESPONSE",
  "alias": "This is a Coil Example",
  "state": "VALIDATED",
  "var_pct": 0,
  "new_read": false,
  "address": 1,
  "new_value": false,
  "at": "2016-07-15T15:11:00.373Z"
}],
"HR": [{
  "value_from": "RESPONSE",
  "published_on": "HI\OFF",
  "name": "HR_1",
  "min": 0,
  "address_offset": 0,
  "max": 100,
  "zero_value": 0,
  "num_value": 89.000000,
  "scaling": 1,
  "alias": "Water Temperature",
  "state": "VALIDATED",
  "var_pct": 17.000000,
  "at": "2016-07-15T15:16:51.407Z",
  "address": 1,
  "units": "Deg C",
  "new_value": true,
  "new_read": true
}, {
  "value_from": "RESPONSE",
  "published_on": "HI\OFF",
  "name": "HR_2",
  "min": -10,
  "address_offset": 0,
  "max": 40,
  "zero_value": 0,
  "num_value": 115.000000,
  "scaling": 1,
  "alias": "Air Temperature",
  "state": "VALIDATED",
  "var_pct": 8.000000,
  "at": "2016-07-15T15:16:51.407Z",
  "address": 2,
  "units": "Deg C",
  "new_value": true,
  "new_read": true
}, {
  "value_from": "RESPONSE",
  "published_on": "HI\OFF",
  "name": "HR_3",
  "min": 0,
  "address_offset": 0,
  "max": 90,
  "zero_value": 0,
  "num_value": 31147.000000,
  "scaling": 1,
  "alias": "Humidity",
  "state": "VALIDATED",
  "var_pct": 86.000000,
  "at": "2016-07-15T15:16:51.407Z",
  "address": 3,
  "units": "%",
  "new_value": true,
  "new_read": true
}],
"IS": [{
  "published_on": "HI\OFF",
  "name": "Input Power Invertor",

```

```

        "str_value": "On",
        "num_value": 1,
        "value_from": "RESPONSE",
        "alias": "IPV",
        "state": "VALIDATED",
        "var_pct": 100,
        "new_read": false,
        "address": 6,
        "new_value": false,
        "at": "2016-07-15T15:12:37.350Z"
    }, {
        "published_on": "HI\OFF",
        "name": "Battery Status",
        "str_value": "Good",
        "num_value": 0,
        "value_from": "RESPONSE",
        "alias": "BS",
        "state": "VALIDATED",
        "var_pct": 100.000000,
        "new_read": false,
        "address": 7,
        "new_value": false,
        "at": "2016-07-15T15:12:37.350Z"
    }, {
        "published_on": "HI\OFF",
        "name": "Num Battery Inputs",
        "address_offset": 0,
        "num_value": 13,
        "value_from": "RESPONSE",
        "alias": "Batt_Inputs",
        "state": "VALIDATED",
        "var_pct": 18.000000,
        "new_read": false,
        "address": 55,
        "new_value": false,
        "at": "2016-07-15T15:14:37.395Z"
    }, {
        "published_on": "HI\OFF",
        "name": "Num Battery Outputs",
        "address_offset": 4,
        "num_value": 2,
        "value_from": "RESPONSE",
        "alias": "Batt_Outputs",
        "state": "VALIDATED",
        "var_pct": 0,
        "new_read": false,
        "address": 55,
        "new_value": false,
        "at": "2016-07-15T15:14:37.395Z"
    }, {
        "published_on": "HI\OFF",
        "name": "Num Indicator Outputs",
        "address_offset": 8,
        "num_value": 0,
        "value_from": "RESPONSE",
        "alias": "Indicator_Outputs",
        "state": "VALIDATED",
        "var_pct": 0,
        "new_read": false,
        "address": 55,
        "new_value": false,
        "at": "2016-07-15T15:14:37.395Z"
    }, {
        "value_from": "RESPONSE",
        "published_on": "HI\OFF",
        "name": "Operating Frequency",
        "address_offset": 0,

```

```

        "zero_value": 0,
        "num_value": 1491.000000,
        "scaling": 1,
        "alias": "OpFreq",
        "state": "VALIDATED",
        "var_pct": 69.000000,
        "new_read": false,
        "address": 60,
        "new_value": false,
        "at": "2016-07-15T15:14:37.395Z"
    }
  ],
  "meta": {
    "description": "Slave_1",
    "value_byte_order": "SNo",
    "name": "Power_          "installation_date": "14\07\2016",
    "location": "Test_Rack",
    "address": {
      "DEVID": "6500004",
      "PORTID": 1,
      "SLAVEID": 10,
      "SWMID": 0
    },
    "manufacturer": "NA"
  },
  "type": "ModbusSlave",
  "id": "10_HR_1_HI"
}

```

### 10.3.1.4 THE RANGE PAYLOAD

Threshold: 100    Hysteresis: 10    Range: 2    3

Address	Type	Name	Units	Event	Payload	QOS	Retain	MQTT Topic	Default Topic	
1	HR	HR_1	Deg C	High Threshold	Range	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	<input type="checkbox"/> <input checked="" type="checkbox"/>

With this rule, if the value of Register 1 goes above 100 the data for the registers between address 2 and 3 will be published.

If there has not been any actual data seen on Modbus for a register in this range it will not appear in the payload.

The num\_value in the payload for each of the holding-registers in this range will be the actual value last seen by the SmartSwarm device on the Modbus network.

```

{
  "model": {
    "state": {
      "HR": [{
        "value_from": "RESPONSE",
        "published_on": "HI\ON",
        "name": "HR_2",
        "min": -10,
        "address_offset": 0,
        "max": 40,
        "zero_value": 0,

```

```

        "num_value": 1762.000000,
        "scaling": 1,
        "alias": "Air Temperature",
        "state": "VALIDATED",
        "var_pct": 0,
        "at": "2016-07-14T19:41:04.109Z",
        "address": 2,
        "units": "Deg C",
        "new_value": false,
        "new_read": true
    }, {
        "value_from": "RESPONSE",
        "published_on": "HI\ON",
        "name": "HR_3",
        "min": 0,
        "address_offset": 0,
        "zero_value": 0,
        "num_value": 0.000000,
        "scaling": 1,
        "alias": "Humidity",
        "state": "VALIDATED",
        "var_pct": 0,
        "at": "2016-07-14T19:41:04.109Z",
        "address": 3,
        "units": "%",
        "new_value": false,
        "new_read": true
    }
  ],
  "meta": {
    "description": "Slave_1",
    "value_byte_order": "SNo",
    "name": "Power_Meter",
    "installation_date": "14\07\2016",
    "location": "Test_Rack",
    "address": {
      "DEVID": "6500004",
      "PORTID": 1,
      "SLAVEID": 10,
      "SWMID": 0
    },
    "manufacturer": "NA"
  },
  "type": "ModbusSlave",
  "id": "10_HR_1_HI"
}

```

## 10.4 TOPICS (HOW)

After we have decided what data we want to publish we can choose a topic on which to publish this data. Clients can then subscribe to this topic in order to see the data.

Two independent topic spaces can be supported simultaneously.

- The custom MQTT Topic space is completely configurable. Use topic strings that make sense for your application-specific MQTT clients and data consumers.
- The Default Topic space is fixed. It uses a well-defined hierarchy which can be mined for data in an application-independent way.

For every individual rule, the custom MQTT Topic, the Default Topic, or both, may be disabled.



*If you are using cellular for the uplink MQTT connection, you are probably limited by bandwidth and/or monthly data usage. The default behavior of the SmartSwarm 351 is for every Rule to publish on both the custom topic and the default topic. You need to profile the cost of these publishes, and selectively enable only those that are required for your use case.*

### 10.4.1 CUSTOM TOPIC SPACE

For every rule in the “Rules and Topics” tab, the “MQTT Topic” field is completely customizable. Any string may be entered. A forward slash will indicate a new level in the topic hierarchy.

For any newly-created rule, the MQTT Topic is pre-filled with strings from the Meta tab for that slave, in the order:  
 <Location>/<Description>/<Name>

For example, given the following slave meta data:

Meta							
Description	Install Date	Location	Manufacturer	Name	Product Code	Byte Order	Version
Slave 1		Test Rack		Power Meter		No Swap	

... the MQTT Topic string will be “Test\_Rack/Slave\_1/Power\_Meter”, as shown:

Dashboard > Devices > Manage Device > Settings > Slave

Exit Editor

Meta										
Address	Type	Name	Units	Event	Payload	QOS	Retain	MQTT Topic	Default Topic	
0	HR	HR_0		None	HR	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
1	HR	HR_1		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
0	IR	IR_0		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
0	CS	CS_0		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +
0	IS	IS_0		None	Default	Exactly Once	<input type="checkbox"/>	Test_Rack/Slave_1/Power_Meter	<input checked="" type="checkbox"/>	- +

Register 0 HR

Notes:

- Any spaces in the Meta data fields will be replaced by underscores in the topic string;
- Forward slashes may be used in the Meta data fields to automatically introduce more levels in the topic space hierarchy.

- If the slave Meta data is changed at some point in time the MQTT Topic string will NOT be automatically updated for any existing rules. This is to prevent problems for any MQTT clients which may have already been configured to subscribe to the old topic string.
- The pre-filled MQTT Topic field can be overwritten by the user at any time.

For every individual rule, MQTT publishes on the custom topic can be disabled by specifying an empty string in the “MQTT Topic” field.

### 10.4.2 DEFAULT TOPIC SPACE

For every individual rule, MQTT publishes on the default topic can be disabled by un-ticking the checkbox for “Default Topic”.

The default topic string is *always* of the form:

<Swarm\_ID>/<Device\_ID>/<Port\_ID>/<Slave\_ID>/<RegisterType>/<Event>

For example, a typical topic string is “0/700000/1/1/HR/SCHEDULING”.

Field	Description
Swarm_ID	Always ‘0’ for devices which are not part of a Swarm.
Device_ID	The serial number of the device, as printed on the hardware label. If the “Device ID” in SmartWorx Hub appears as “203-01-6200799”, then the Device_ID field is just the last 7 digits: “6200799”.
Port_ID	0 = RS-232 1 = RS-485
Slave_ID	The address of the Modbus Slave which is providing the payload information.
RegisterType	CS: Coil IS: Discrete Input IR: Input Register HR: Holding Register
Event	All Discrete Input registers on this slave will be published. Read: “READ” Change: “CHANGE” Delta: “DELTA” High Threshold: “HI” Low Threshold: “LO” High Rate: “RATE-HI” Low Rate: “RATE-LO” Scheduled: “SCHEDULING” Global Read: “READ” Global Change: “CHANGE”

Table 38. The Default Topic

One extra field is appended to the default topic string in certain situations:

If the Event is one of the STATEFUL types, then a State value of "OFF" or "ON" is appended. (i.e. For events of type High Threshold, Low Threshold, High Rate, Low Rate.)

Because the default topic string can NOT be changed by the user, you can rely on the topic space hierarchy to be consistent across all devices, slaves, and rules. For example:

To subscribe to:	Use this topic string:
All messages from device 7000000	0/7000000/#
All "alarm" activations from device 7000000	0/7000000/+/+/+/ON

Table 39. Default Topic example

### 10.4.2.1 PAYLOAD FORMAT FOR DEFAULT TOPICS

Publishes on the default topic space differ from publishes on the custom topic space in two important respects:

- 1) Publishes on the default topic space are always SERIALIZED at the register level. (i.e. If the Payload selection is more than one register, then there will be one MQTT message published on the custom topic, but multiple MQTT messages published on the default topic.)
- 2) Every publish on the default topic space is accompanied by another message containing "meta" data for the slave. (If the publish involves multiple registers, there is still only one meta message.)

### 10.4.2.2 META MESSAGES

The meta message is automatically published on the topic string:

<Swarm\_ID>/<Device\_ID>/<Port\_ID>/<Slave\_ID>/meta

The payload contains the information from the "Meta" tab for that slave on SmartWorx Hub.

For example, given the following Slave information:

ADVANTECH B+B SMARTWORX Device ID: 203-01-6200799 Slave 1: RS485 Help Log off  
Hello, admin

Dashboard Devices Users Technology Providers Configuration Profiles Manufacturing Password Contact

Dashboard > Devices > Manage Device > Settings > Slave

Save Push to Device Exit Editor

Meta Inputs (1x) Coils (0x) Input Registers (3x) Holding Registers (4x) Rules and Topics

Description	Install Date	Location	Manufacturer	Name	Product Code	Byte Order	Version
Warehouse heaters	29 Mar 2016	Test Rack	Carlo Gavazzi	EM24	EM24.DIN.AV9.3.X.15.X	Swap Words only	1.0

... the meta message will look like this:

```
"0/6200799/1/1/meta": {
  "address": {
    "DEVID": "6200799",
    "PORTID": 1,
```

```
        "SLAVEID": 1,  
        "SWMID": 0  
    },  
    "description": "Warehouse heaters",  
    "installation_date": "29/03/2016",  
    "location": "Test Rack",  
    "manufacturer": "Carlo Gavazzi",  
    "name": "EM24",  
    "product_code": "EM24.DIN.AV9.3.X.IS.X",  
value_byte_order": "Sword",  
    "version": "1.0"  
}
```

**11. VERIFY YOUR DATA FLOW**

In order to verify your data flow, we recommend that you verify each step below.

Step	What to verify	Reference
1	Verify your physical connection to the Modbus bus.	Section 4.5 Appendix 3
2	Verify that your SmartSwarm device has a secure connection to SmartWorx Hub.  A secure connection to SmartWorx Hub exists if the USR LED, on the front panel of the SmartSwarm device, is ON (yellow).	Section 4.3 and 4.4 Section 5.1 to 5.4 Appendix 3
3	Verify that your Modbus interface is configured to the correct settings. The settings you select must match those of the Modbus Master.	Section 7 Appendix 3
4	Verify that your MQTT interface is configured to the correct settings. The settings you select must match those of the MQTT broker you wish to publish to.  We recommend that you use non-secure settings until you have verified connectivity. Then enable a fully secure connection.  (e.g. Initially, get some test-data published on a public MQTT broker, without enabling TLS to secure the transport layer.)  Once you have verified your data flow using non-secured sample data, we recommend that you secure your entire data flow.	Section 8 Appendix 6
5	Verify your Slave enrichment.	Section 9
6	Verify that you have configured the Event Rules correctly. Data will only be published in accordance with the Event Rules that you have enabled.	Section 10 (Events - WHEN) Appendix 3
7	Verify that the data you will publish is the data you intend to publish.	Section 10 (Payloads - WHAT) Appendix 3

8	Verify the MQTT Topic that your events will be published on.	Section 10 (Topics - HOW)  Appendix 3
---	--	---

Table 40. Verify your Data Flow

**12. OTHER DOCUMENTATION**

Document Title	Where?
Modbus Serial Line Protocol and Implementation Guide	<a href="http://www.modbus.org/docs/Modbus over serial line V1 02.pdf">http://www.modbus.org/docs/Modbus over serial line V1 02.pdf</a>
MQTT and the NIST Cybersecurity Framework”	<a href="http://docs.oasis-open.org/mqtt/mqtt-nist-cybersecurity">http://docs.oasis-open.org/mqtt/mqtt-nist-cybersecurity</a>
SmartWorx Hub User Manual	<a href="https://hub.bb-smartworx.com/Login/Help?HelpFile=bbdms_help.pdf">https://hub.bb-smartworx.com/Login/Help?HelpFile=bbdms_help.pdf</a>
OpenVPN documentation	<a href="https://openvpn.net/index.php/open-source/documentation/howto.html#client">https://openvpn.net/index.php/open-source/documentation/howto.html#client</a>

Table 41. Other Documentation

**13. APPENDIX 1 - HARDWARE RATINGS**

**13.1 ENVIRONMENTAL**

IoT Gateway SmartSwarm 300		
Temperature range	Operating Storage	-40 to +75 deg.C -40 to +85 deg.C
Cold start	-35 deg. C -40 deg. C	Data transfers via mobile network are available immediately Data transfers via mobile network are available approximately in five minutes after the start of the device. Everything else is functional immediately.
Humidity	Operating Storage	0 to 95 % relative humidity non condensing 0 to 95 % relative humidity non condensing
Altitude	Operating	2000 m / 70 kPa
Degree of protection		IP42
Supply voltage		10 to 60 V DC
Consumption	Idle Average Peak	2,5 W 4W 11 W
Dimensions		55x97x125 mm (DIN 35 mm)
Weight		Approximately 400 g (depends on interface)

Antenna connectors		2 x SMA – 50 Ohm
User interface	2x ETH USB I/O RS-485 RS-232	Ethernet (10/100 Mbit/s) USB 2.0 (not currently supported) 6-pin panel socket (not currently supported) 4 pin panel socket 5 pin panel socket

Table 42. Environmental

### 13.2 TYPE TESTS

Table 43 Type Tests

Phenomena	Test	Description	Test Levels
ESD	EN 61000-4-2	Enclosure contact	± 6 kV (crit. A)
		Enclosure air	± 8 kV (crit. A)
RF field AM modulated	IEC 61000-4-3	Enclosure	20 V/m (crit. A) (80 – 2700 MHz)
Fast transient	EN 61000-4-4	Signal ports	± 2 kV (crit. A)
		Power ports	± 2 kV (crit. A)
		Ethernet ports	± 2 kV (crit. A)
Surge	EN 61000-4-5	Ethernet ports	± 2 kV (crit. B), shielded cable
		Power ports	± 0,5 kV (crit. B)
		I/O ports	±1kV,LtoL(crit. A) ±2kV,LtoGND(crit. A)
RF conducted	EN 61000-4-6	All ports	10 V/m (crit. A) (0,15 – 80 MHz)
Radiated emission	EN 55022	Enclosure	Class B
Conducted emission	EN 55022	DC power ports	Class B
		Ethernet ports	Class B
Power frequency magnetic field	EN 61000-4-8	Enclosure	160 A/m (crit. A)
Dry heat	EN 60068-2-2	+75 °C, 40 % rel. humidity	
Cold	EN 60068-2-1	-40 °C	
Dump heat	EN 60068-2-78	95 % rel. humidity (+40 °C)	

Table 44. Type Tests

### 13.3 CELLULAR MODULE

LTE module for EMEA	
LTE parameters	Bit rate 100 Mbps (DL) / 50 Mbps (UL) 3GPP rel. 8 standard Supported bandwidths: 5 MHz, 10 MHz, 20 MHz Supported frequencies: 800 / 900 / 1800 / 2100 / 2600 MHz
HSPA+ parameters	Bit rate 21,1 Mbps (DL) / 5,76 Mbps (UL) 3GPP rel. 7 standard UE CAT. 1 to 6, 8, 10, 12, 14 3GPP data compression Supported frequencies: 900 / 2100 MHz
UMTS parameters	PS bit rate 384 kbps (DL) / 384 kbps (UL) CS bit rate 64 kbps (DL) / 64 kbps (UL) W-CDMA FDD standard Supported frequencies: 900 / 2100 MHz
GPRS/EDGE parameters	Bit rate 237 kbps (DL) / 59,2 kbps (UL) GPRS multislots class 10, CS 1 to 4 EDGE multislots class 12, CS 1 to 4, MCS 1 to 9 Supported frequencies: 900 / 1800 / 1900 MHz
Supported GPRS/EDGE power classes	EGSM 900: Class 4 (33 dBm) GSM 1800/1900: Class 1 (30 dBm) EDGE 900: Class E2 (27 dBm) EDGE 1800/1900: Class E2 (26 dBm)

Table 45. Cellular Module

13.4 OTHER TECHNICAL PARAMETERS

Other technical parameters	
CPU power	2 DMIPS per MHz
Flash memory	256 MB
RAM	512 MB
M-RAM	128 kB

Table 46. Technical Parameters

14. APPENDIX 2 - GENERAL SETTINGS

14.1 CONFIGURABLE ITEMS

For every SmartSwarm device, there are some general settings and options that are available to you.

**Manage Device** Hello, pconway@adv

Dashboard Devices Users Technology Providers Configuration Profiles Password Contact

Dashboard > Devices > Manage Device

Device ID: 203-01-6500003

Name: PaulC Factory Systems

Status: Operational

Firmware: 0.4.9

DeviceType: SG30300322-51

Online

Save Cancel Push Firmware History **Settings** Add/Upgrade Apps

**Manage Apps**

Remove Selected

Name	Tag	Type	Version	Help	Added
<input type="checkbox"/> Modbus2MQTT	Modbus2MQTT	Application	0.4.9	<a href="#">Help</a>	5/24/2016 11:39:29 AM

14.1.1 SETTINGS

The Network settings enable you to configure the operation of the ETH ports and the Cellular interface of your device.

By default, ETH0 has a static IP address of 192.168.1.1.

By default, ETH0 runs a DHCP server, which will serves a DHCP address to a connecting device. This means that you should configure your desktop/laptop to take an IP address automatically when you connect it to ETH0 of the SmartSwarm device.

There is a local web-server, for local configuration purposes, served on ETH0 (<http://192.168.1.1>).

We recommend that you do not change the ETH0 default settings.

By default, ETH1 runs as a DHCP client.

By default, the cellular interface is not configured. But note that you may have previously configured the Cellular Interface locally on your device.



*Changing network settings from SmartWorx Hub can result in breaking the working secure connection your device has to SmartWorx Hub.  
Please ensure you are applying appropriate network settings to your device, or that you have a contingency plan (e.g. local device access is available) in the event that you unintentionally cause the secure connection to drop.*

**Settings**

[Help](#) [Log off](#)

Hello, pconway@advantech-bb.com

Dashboard
Devices ▾
Users ▾
Technology Providers
Configuration Profiles
Password ▾
Contact

Dashboard > Devices > Manage Device > Settings

**Network**

DHCP

OpenVPN

NTPClient

### Device Settings

**Device Name** PaulC Factory Systems

\* Required Field

Advanced Settings +

Network

**ETH0 (LAN)** -

Protocol:

IP Address:

Network Mask:

Gateway :

DNS Server(s) :

**ETH1 (WAN)** -

**Cellular** -

### 14.1.2 DHCP

The DHCP settings apply only to the DHCP server that runs on ETH0.



At the time of writing, it is not possible to turn off the DHCP server that runs on ETH0. Please be careful not to connect ETH0 of the device into a LAN port that is also serving DHCP addresses.

### 14.1.3 OPENVPN

You may configure up to 2 OpenVPN tunnels to run on your device.

This may be useful if you need the ability to reach the local-web-server on the device -- remotely, for example.

The screenshot shows the 'Settings' page for a device named 'PaulC Factory Systems'. The left sidebar has a menu with 'OpenVPN' circled in red. The main content area is titled 'Device Settings' and includes a 'Cancel' and 'Apply changes' button. Below this is an 'Advanced Settings' section with a plus sign. Underneath is an 'OpenVPN' section containing a 'VPN Tunnel 1' configuration form. The form includes the following fields:

- Enable Tunnel:
- Protocol: TCP (dropdown)
- VPN Server(IP Port): 148.251.6.41 1194 (text input)
- Local Port: 1194 (text input)
- Verbosity: 3 (dropdown)
- Use LZ0 Compression: Yes (dropdown)
- Client Mode:

Below the form are three expandable sections: 'CA Certificate', 'Client Certificate', and 'Key'. At the bottom, there is a 'VPN Tunnel 2' section which is currently collapsed.

The user interface enables you to configure an OpenVPN tunnel to an OpenVPN server.

Before you begin to use an OpenVPN service, we recommend that you are familiar with the OpenVPN documentation, which is available here:

<https://openvpn.net/index.php/open-source/documentation/howto.html#client>

OpenVPN	
Enable Tunnel	Enable or Disable this tunnel interface. Disabled by default.
Protocol	UDP or TCP (TCP is default)
VPN Server (IP Port)	The IP Address of the OpenVPN Server, and the port the Server is listening on. This must be entered as a single string, like in this example: 148.251.6.41 1194
Local Port	The local Port the device will (optionally) use to bind to the OpenVPN service on the server
Verbosity	Enable the debug-message level you want on your Device. The bigger the number, the more debug messages are written into the OpenVPN message log. We recommend that you use 0 here.
LZO Compression	Enable or Disable compression on the OpenVPN client-server connection. If compression is enabled on the server it must also be enabled on the device. Enabled by default.
Client Mode	Enabled or Disabled. Enabled by default. Must be enabled if the Tunnel is enabled.
CA Certificate	The Certification Authority's certificate, which is used to generate the Client Certificate from the Certification Request generated by the Private Key.  This must be the same CA certificate (or be in the chain-of-trust) that is used by the Server.  The CA Certificate is the Server's Public Key.
Client Certificate	The Client Certificate is the certificate created by the CA for the Client (Device), from the Certificate Request that was sent to the CA.  The Client Certificate is the Device's Public Key.

Key	The Private Key (for the Device) that is used to generate the Certification Request. The Certification Request is what you send to the Certification Authority.
-----	--

Table 47. OpenVPN fields

When OpenVPN feature is enabled, the Client Key, the Client Certificate, and the CA Certificate will be sent to the Device.

When the OpenVPN feature is disabled, all of these items will be removed from the Device.

So how do you create your Key, how do you get your Client Certificate, and how do you know what the CA certificate is?

You can generate your own private key (intended to be the Private Key of the Device).

Please consult “openssl” documentation, and please refer to your OpenVPN server’s documentation.

Here’s an example of how to create a private key. (There are many options that you can apply here; we’re using one option for illustration purposes only):

```
$ openssl genrsa -out MyDevicePrivate.key 2048
```

You now have the “Key” required.

Next, you need to generate a Certificate Signing Request. Here’s an example (again, this is only one of many possible examples):

```
$ openssl req -new -sha256 -key MyDevicePrivate.key -out
CertificateRequest.csr
```

```
Country Name: <your 2 letter country code>
State or Province Name: <your province name>
Locality Name: <your location name>
Organization Name: <your organization name>
Organizational Unit Name: <your team name>
Common Name: <your domain name> (e.g. "devid6500003")
email: <your email>
Challenge password: <blank, press enter>
Optional company name: <blank, press enter>
```

The output from this sequence is a file named “CertificateRequest.csr”.

Now, you must send this Certificate Signing Request to your Certificate Authority for signing.

The CA that signs this certificate must be the same CA, or in the chain-of-trust of the CA, that has signed the Server’s Certificate.

You will receive back your signed certificate (this is the Client Certificate that you require), along with the server’s CA certificate (this is the CA Certificate that you require).

14.1.4 NTP CLIENT

You may specify up to 4 network time protocol servers for this Device.

**Settings** Help Log off  
Hello, pconway@advantech-bb.com

Dashboard Devices Users Technology Providers Configuration Profiles Password Contact

Dashboard > Devices > Manage Device > Settings

Network  
DHCP  
OpenVPN  
**NTPClient**

**Device Settings**

Device Name PaulC Factory Systems  
Cancel Apply changes

\* Required Field

Advanced Settings +

NTPClient

**NTP Servers** -

Server 1:

Server 2:

Server 3:

Server 4:

14.2 NON-CONFIGURABLE ITEMS

14.2.1 FIREWALL

There is a built-in Firewall on your SmartSwarm device.

This firewall cannot be re-configured by the user.

The default Firewall policy is to drop all “input” and “forward” requests, and to accept all “output” requests.

The following Firewall exception rules are then applied to the policy for incoming requests:

Interface	DHCP server	ICMP (ping)	HTTP	SSH	Forward to internet
ETH0	✓	✓	✓	✓	
ETH1		✓	✓	✓	✓
Cellular					✓

Tunnel*		✓	✓	✓	
---------	--	---	---	---	--

**Table 48. Firewall rules**

Note that some Firewall exception rules will be applied automatically, depending upon whether you have configured OpenVPN. For example, the \*Tunnel interface will only exist when you have enabled an OpenVPN tunnel.

**15. APPENDIX 3 - DIAGNOSTICS AND TROUBLESHOOTING**

There is a local web-server interface on ETH0 of the SmartSwarm device.

This interface is intended to be used for two purposes:

- a) Configure the device's outbound (WAN) connectivity (using either the Cellular interface, or ETH1).
- b) Diagnosing and Troubleshooting problems, in collaboration with the Advantech B+B SmartWorx technical support team.

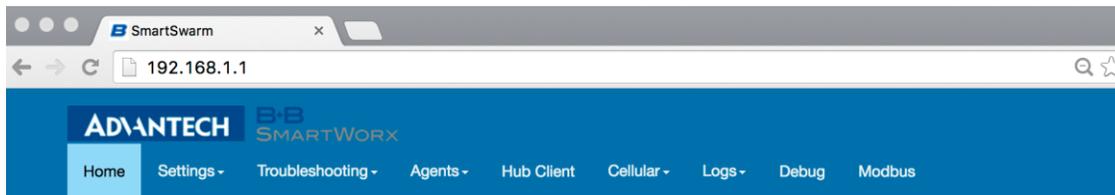
**15.1 THE LOCAL WEB INTERFACE**

There is an embedded web-server which provides a local interface on ETH0.

By default, ETH0 of the device is configured with IP address 192.168.1.1, subnet 255.255.255.0.

ETH0 is configured as a DHCP server: This means that if you physically connect ETH0 to your laptop/desktop the device will automatically serve an IP address of 192.168.1.x to your laptop/desktop.

The local web interface looks like this:

**SmartSwarm Local WebServer**

This page can be used to configure and diagnostic device's configuration

Below you can find some useful information

System Information	
Firmware Version	0.4.9
Components Version	0.4.9
Serial Number	6500004
U-Boot Version	U-Boot 2014.04 (Feb 12 2016 - 14:08:11)
Uptime	15:34:31 up 1 min, 0 users, load average: 7.97, 1.90, 0.63

There are nine Tabs: Home; Settings; Troubleshooting; Agents; Hub Client; Cellular; Logs; Debug and Modbus

---

### 15.1.1 HOME

From the Home tab, you can see some important information about your SmartSwarm device:

- Firmware Version
- Components Version
- Serial number
- U-Boot Version
- Device uptime, connected users, load average

---

### 15.1.2 SETTINGS

The Settings tab enables you to configure your connectivity ports:

- Cellular
- ETH0
- ETH1

If you intend to use the Cellular interface for your outbound connection you must enter your APN and network credentials here.

By default, ETH0 will operate as a LAN interface only and ETH1 will expect to be served an address from a DHCP server.

We assume that the DHCP server that serves this address will also provide a route to the internet.

If this is not the case, you may need to re-configure your ETH1 interface.

---

### 15.1.3 TROUBLESHOOTING

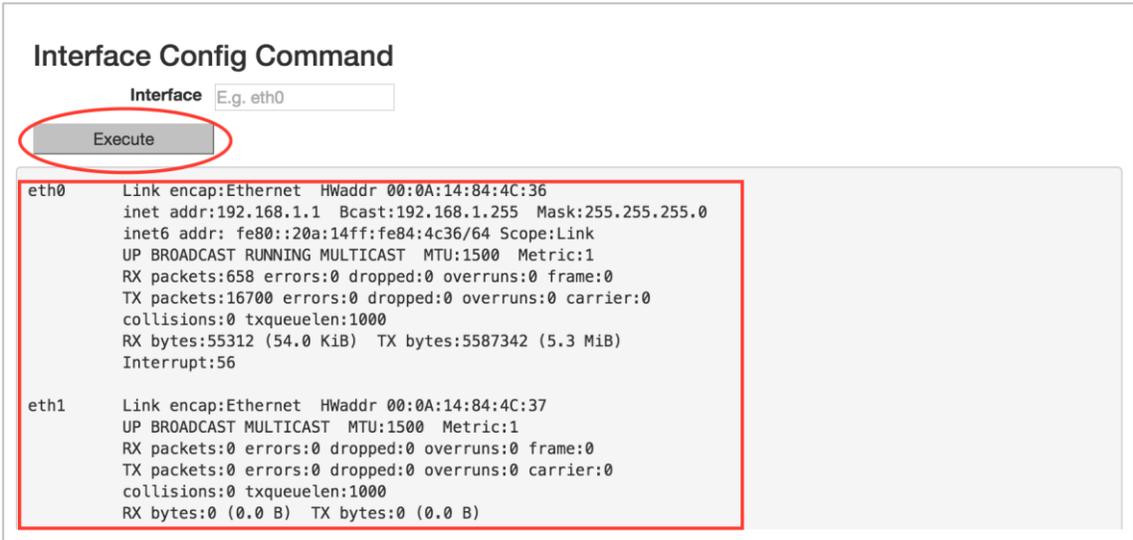
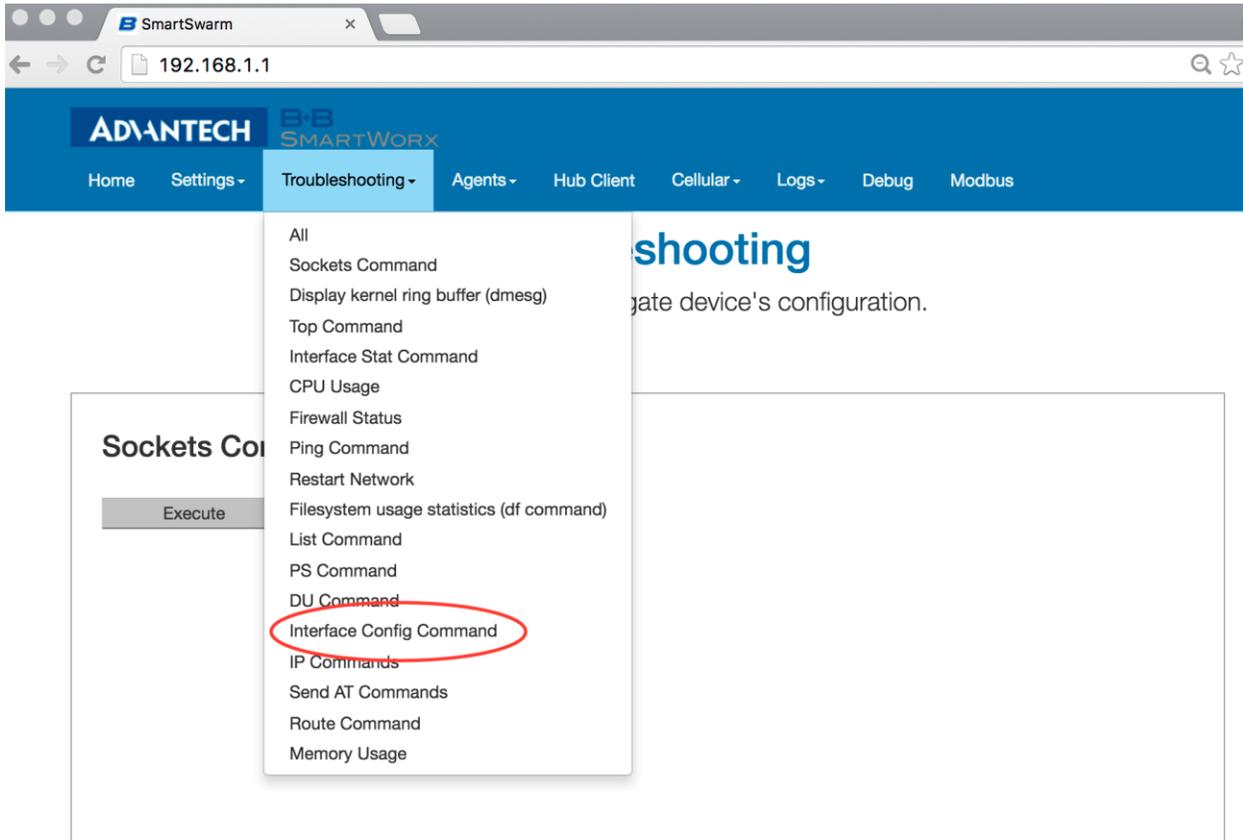
The Troubleshooting tab gives you the ability to see the actual internal device status of a number of key interfaces, processes and settings.

This interface gives you a drop-down list of commands that you can trigger, so that you can gather some potentially valuable run-time information. In the case your device is not performing as you think it should.

When you're working with the Advantech B+B SmartWorx technical support engineer, he may ask you for some of the details that are available from this Tab.

In most cases, you must select the command from the drop-down list, then hit the '**Execute**' button.

This will execute the command on the device, and feedback the results to the browser window.



### 15.1.4 HUB CLIENT

Using this tab, you can change the default SmartWorx Hub Server instance that your device connects. By default, your device will connect to hub.bb-smartworx.com using https on port 443.

If, for example, you have a hosted instance of SmartWorx Hub, you can change your devices' settings to connect to your hosted instance instead.

### 15.1.5 CELLULAR

Use the Cellular tab to get some cellular integrity diagnostics from your device.

Using this tab you can get:

- Signal Strength
- System Information
- Signal Information
- Card Status

The screenshot shows a web browser at the address 192.168.1.1. The header includes the ADVANTECH B+B SMARTWORX logo and a navigation menu with items: Home, Settings, Troubleshooting, Agents, Hub Client, Cellular, Logs, Debug, and Modbus. The 'Cellular' menu is open, showing options: All, Signal Strength, System Information, Signal Information, and Card Status. Below the menu, a text box says 'Use this page to investigate d...figuration.' The main content area displays the 'Signal Strength' diagnostic page, which has an 'Execute' button and a terminal window showing the following output:

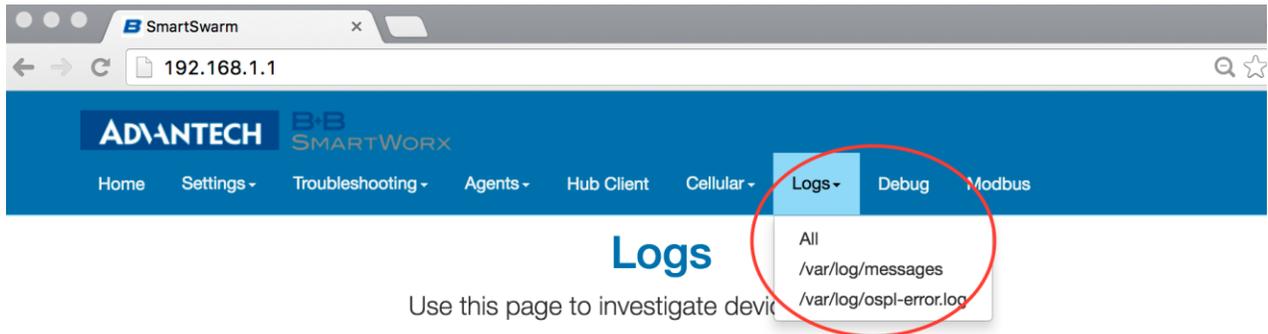
```
[/dev/cdc-wdm0] Successfully got signal strength
Current:
  Network 'umts': '-82 dBm'
Other:
  Network 'cdma-1xevd0': '-125 dBm'
RSSI:
  Network 'umts': '-82 dBm'
  Network 'cdma-1xevd0': '-125 dBm'
ECIO:
```

### 15.1.6 LOGS

The SmartSwarm device will keep debug message logs internally.

During the troubleshooting session it may be important to open the Logs tab, and to take a copy of the messages from one of the debug-logs available.

To see the logs you must turn on “follow” mode and Execute.



### 15.1.7 MODBUS

The Modbus Tab will draw a graph of the Modbus Slaves that this Device knows about.

This is a graphical representation of the Slave Maps currently residing on this device.

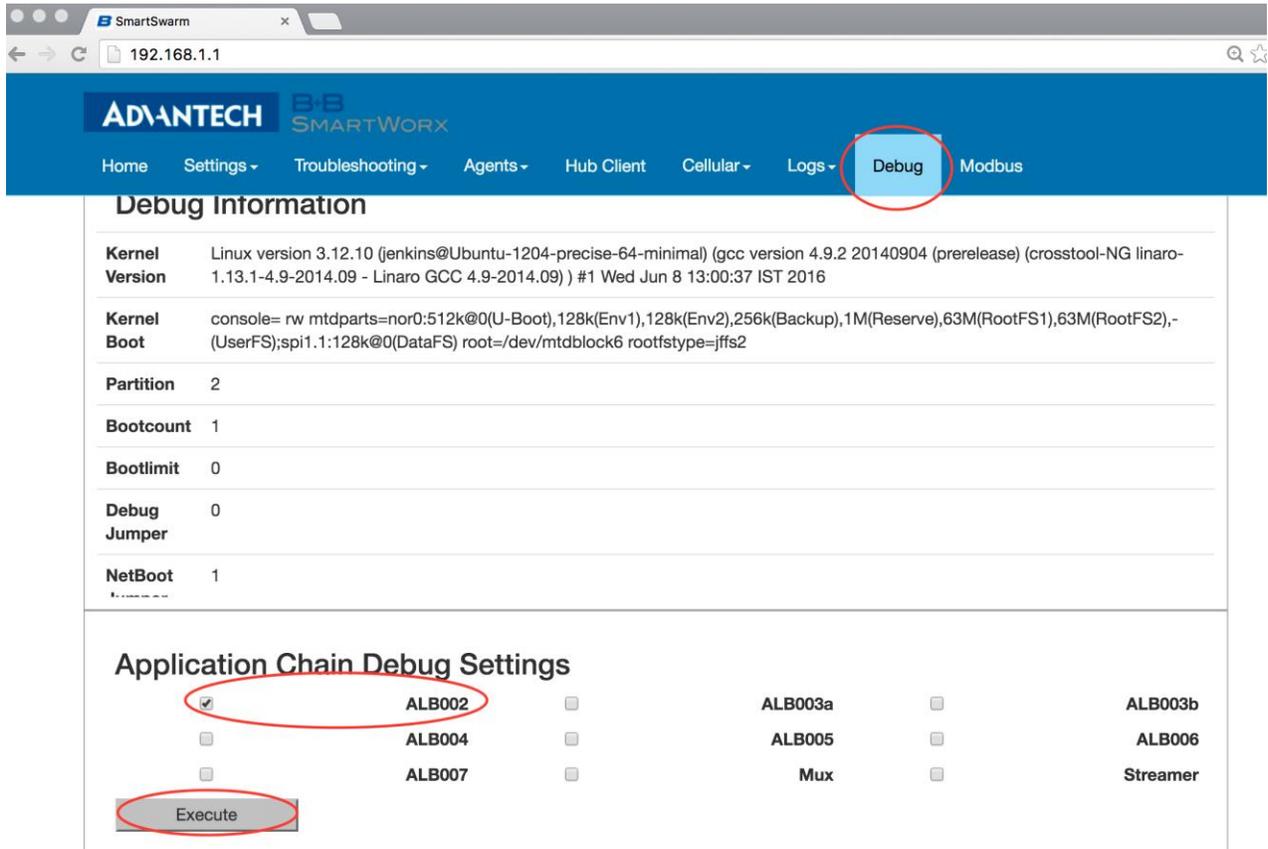
Note that this representation will include any Slave devices that were known on this device, even if they are not currently active.

The screenshot shows the 'Modbus Network' interface in a web browser. The browser address bar shows '192.168.1.1'. The navigation menu includes Home, Settings, Troubleshooting, Agents, Hub Client, Cellular, Logs, Debug, and Modbus. The 'Modbus Network' section has an 'Execute' button and a tree diagram. The tree starts with a root node '6500004 RS485'. It branches into two nodes: '1' and '2'. Node '1' is connected to 'IR', which lists registers 29-34: Nominal Input Current, Nominal Input Frequency, Nominal Output Frequency, Nominal Power Factor, and Nominal Battery Voltage. Node '2' is connected to 'IS', which lists registers 2, 43, 46, 49, 72, 73, 80, 81, 127, and 128: Automatic Battery Test Enabled, Battery Charger on, Inverter Ready, Power Factor Correction State, Load On Inverter, Bypass Active, Replace Battery, Battery Under Test, Load On Battery, and Load On Bypass. Below node '2', there are also registers 3, 9, and 10: Number of Output Lines, Number of Power Modules, and Number of Battery Modules.

### 15.1.8 DEBUG AND AGENTS

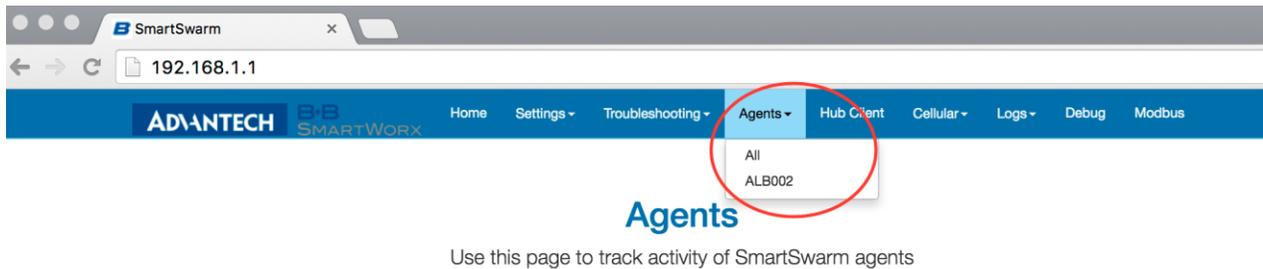
It's best to use the Debug and Agents Tabs in conjunction with each other.

In the Debug Tab you can see some static debug information and you can select which Agent(s) you wish to see run-time information from.



In the Agents tab you can see run-time information (output) from the enabled Application Agents.

Note that, even if an Agent has been “enabled” on the Debug Tab, it will only appear on the Agents Tab if there is data actually being published by that Agent.



In the screenshots shown here we have enabled the ALB002 Agent for debug purposes.

You can use the ALB002 Agent to verify that your Serial Interface is configured correctly. If you enable Debug on Agent ALB002, and your serial interface settings are correct, you should see data streaming for ALB002 on the Agents Tab.

The default setting for debugging each of the Agents is 'disabled'. We recommend you leave these settings disabled (not checked), before and after your Troubleshooting session. This is because the more debugging you enable, the more the performance of your device will degrade.

We recommend that you only use the Debug interface, if necessary, in collaboration with the Advantech B+B SmartWorx technical support personnel.

### 15.1.9 TSED

The Time Series Event Detection Agent is treated differently than the other Application Agents. This is the agent that will detect when complex trigger-events occur: i.e. when the data-pattern detected matches a defined Event Rule.

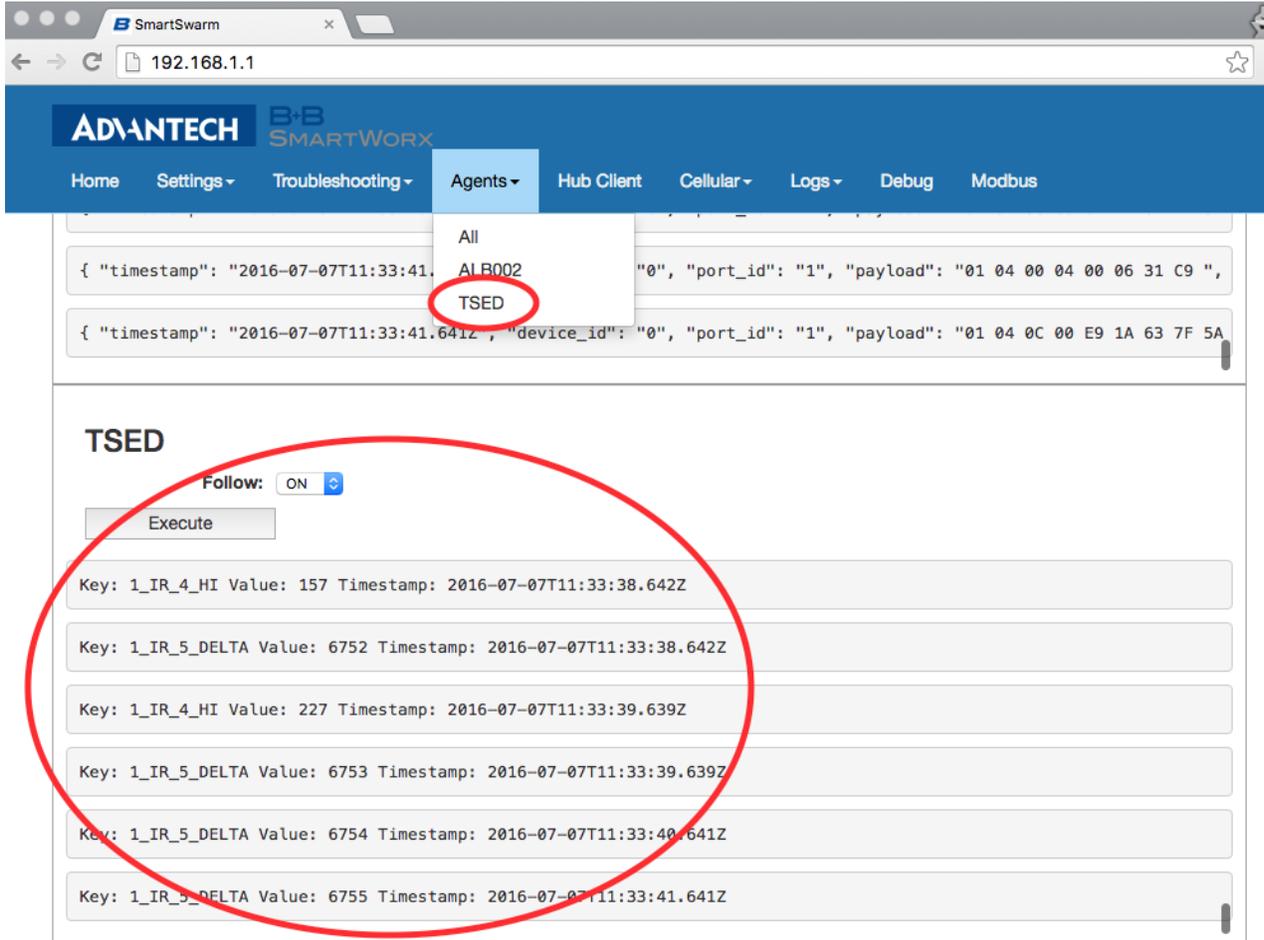
You cannot turn this Agent debug stream on or off.

When there are "complex" Events causing Triggers, the TSED agent will always be available. It will stream output to the Web Server's "Agents" tab.

The following Events are Complex Events: Delta; High Threshold; Low Threshold; High Rate; Low Rate.

This is provided so that you can verify your data-flow and data-enrichment process. If you see data streaming through the TSED interface you will know that your complex Events are triggering successfully, and there will be data published to the MQTT interface.

NOTE: you will only see TSED as an available Agent when complex events are being triggered.



16. APPENDIX 4 - SLAVE MAP FORMATS

Maps may be imported in JSON or Microsoft Excel formats. These files are available for download from SmartWorx Hub.

16.1 EXCEL

Excel can be used to create slave maps. Download the map template file directly from SmartWorx Hub.

Most Modbus control systems allow the export of slave and register data to csv format. This can then be copied into the template sheet.

The data from the Modbus control system must be manipulated into one of the formats expected by the SmartWorx Hub import facility.

The template sheet follows the structure of the Slave Map page on SmartWorx Hub.

See section entitled "Editing Slaves" for more information on entry fields.

The notation in the Excel sheet corresponds to the notation on the SmartWorx Hub editor tabs like this:

Excel Sheet Tab	SmartWorx Hub Tab	Notes
meta	Meta	Meta data for this Modbus Slave.
address	<none>	Provides information about the Modbus Slave (e.g. the Modbus Slave address), and information about the SmartSwarm device (device id, port id, swarm id).
CS	Coils (0x)	The Modbus Coils for this Slave
IR	Input Registers (3x)	The Modbus Input Registers for this Slave
HR	Holding Registers (4x)	The Modbus Holding Registers for this Slave
IS	Inputs (1x)	The Modbus Inputs for this Slave

Table 49. Excel Sheet tabs

Example of the Address Tab:

	A	B	C	D	E	F	G
1	DEVID	PORTID	SLAVEID	SWMID			
2	700000	1	2	0			
3							
4							
5							
6							
7							

meta address CS IR HR IS

Item	Description
DEVID	Serial number of the SmartSwarm 351 device which can be found on the device label
PORTID	0 when interface is RS-232 1 when interface is RS-485
SLAVEID	The number of the slave (1 - 247)
SWMID	This is always 0

Table 50. Excel sheet, Address tab

Example of the Meta Tab:

	A	B	C	D	E	F	G	H
1	description	installation_date	location	manufacturer	name	product_code	value_byte_order	version
2	VSD	14-Mar-16	Library	Emerson	AHU-01 SAF	Commander SK	SNo	1.00
3								
4								
5								
6								
7								
8								

Example of the HR (Holding Registers) Tab:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	address	address_offset	name	alias	datatype	length	zero_value	max	min	scaling	units	num	val
2	614	0	Drive Enable		ENUM	1						0	OFF
3	614	0	Drive Enable		ENUM	1						1	ON
4	500	0	Motor Frequency		UINT16	16	0			10	Hz		
5	806	0	Status relay state		ENUM	1						0	OFF
6	806	0	Status relay state		ENUM	1						1	ON
7	0	0	Minimum set speed		UINT16	16	0	550	0	1			
8	624	0	Energy meter: kWh		UINT16	16	0			1000	kWh		
9	629	0	Sequencing bit: Run forward		UINT16	16	0			1			
10	616	0	Reset Energy Meter		ENUM	1						0	OFF
11	616	0	Reset Energy Meter		ENUM	1						1	ON
12	7	0	Motor rated voltage		UINT16	16	0	690	230	1	v		
13	42	0	Serial comms baud rate		UINT16	16	0			1	bps		
14													
15													

Example of the IR (Input Registers) Tab:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	address	address_offset	name	alias	datatype	length	zero_value	max	min	scaling	units	num	val
2	3	0	Power Consumption		UINT16	16	0			1000	kWh		
3	7	0	Speed		UINT16	16	0				rpm		
4	38	0	Pump Status		ENUM	1							0 Pumped Turned Off
5	38	0	Pump Status		ENUM	1							1 Pumped Turned On
6	38	1	Pump Status		ENUM	1							0 Right Rotation
7	38	1	Pump Status		ENUM	1							1 Left Rotation
8	38	2	Pump Status		ENUM	1							0 Difference <+10%
9	38	2	Pump Status		ENUM	1							1 Difference >+10%
10	38	3	Pump Status		ENUM	1							0 Extern off not active
11	38	3	Pump Status		ENUM	1							1 Extern off active
12	38	4	Pump Status		ENUM	1							0 Single pump
13	38	4	Pump Status		ENUM	1							1 Double pump
14	38	5	Pump Status		ENUM	1							0 Normal Mode
15	38	5	Pump Status		ENUM	1							1 Manual Override
16	38	6	Pump Status		ENUM	1							0 Normal Mode
17	38	6	Pump Status		ENUM	1							1 Q/H values are invalid
18	38	7	Pump Status		ENUM	1							0 Extern min not active
19	38	7	Pump Status		ENUM	1							1 Extern min active
20	38	13	Pump Status		ENUM	1							0 Normal Mode
21	38	13	Pump Status		ENUM	1							1 Wink/Service mode
22													
23													
24													
25													
26													
27													

Example of the CS (Coils) Tab:

	A	B	C	D	E
1	address	name	alias	val0	val1
2	24	Coil Example	JN1	OFF	ON
3					
4					
5					
6					
7					

Example of the IS (Discrete Inputs) Tab:

	A	B	C	D	E
1	address	name	alias	val0	val1
2	30	Input Example	IP1	NO	YES
3					
4					
5					
6					
7					

After completing the sheet, save the sheet with an appropriate filename (e.g. map-2.xlsx).

This file can now be imported into SmartWorx Hub by clicking the **Load Maps** link.

## 16.2 JSON

The alternative import-map format is JSON format.

This section will show some examples of appropriate JSON formatted input maps.

The JSON map below is a minimal example of a slave map with no registers defined.

```
{
  "id": "",
  "model": {
    "meta": {
      "address": {
        "DEVID": "7000000",
        "PORTID": 1,
        "SLAVEID": 1,
        "SWMID": 0
      },
      "description": "",
      "installation_date": "",
      "location": "",
      "manufacturer": "",
      "name": "",
      "product_code": "",
      "value_byte_order": "SNo",
      "version": ""
    },
    "state": {
      "CS": [],
      "HR": [],
      "IR": [],
      "IS": []
    }
  },
  "type": "ModbusSlave"
}
```

To add a Holding register to the map insert the following under the HR section. To add another register, repeat the section separated by a comma.

```
"HR": [
  {
    "address": "500",
    "address_offset": "0",
    "name": "Motor Frequency",
    "datatype": "UINT16",
    "length": "16",
    "zero_value": "0",
    "scaling": "10",
    "units": "Hz",
    "state": "VALIDATED"
  }
]
```

```
    }
  ]
}
```

To add an Input register to the map insert the following under the IR section. To add another register, repeat the section separated by a comma.

```
"IR": [
  {
    "address": "3",
    "name": "Power Consumption",
    "datatype": "UINT16",
    "length": "16",
    "zero_value": "0",
    "scaling": "1000",
    "units": "kWh",
    "state": "VALIDATED"
  }
]
```

To add a Coil to the map insert the following under the CS section. To add another register, repeat the section separated by a comma.

```
"CS": [
  {
    "address": 24,
    "name": "Coil Example",
    "alias": "JN1",
    "state": "VALIDATED",
    "available_values": [
      "OFF",
      "ON"
    ]
  }
]
```

To add a Discrete Input to the map insert the following under the IS section. To add another register repeat the section separated by a comma.

```
"IS": [
  {
    "address": 30,
    "name": "Input Example",

    "alias": "IP1",
    "state": "VALIDATED",
    "available_values": [
      "NO",
      "YES"
    ]
  }
]
```

ENUMs are a special case for Holding and Input registers and are added to the map as follows

```
{
```

```

    "address": 40,
    "address_offset": 2,
    "name": "Pump Command",
    "alias": "",
    "datatype": {
      "enum_type": {
        "num": [
          0,
          1
        ],
        "val": [
          "Normal Operation",
          "Max Speed"
        ]
      }
    },
    "length": 1,
    "zero_value": null,
    "max": null,
    "min": null,
    "scaling": null,
    "units": "",
    "state": "VALIDATED"
  }
}

```

It is recommended that JSON maps be validated using an online JSON formatter. It is difficult to find errors, especially when the maps contain a large number of registers.

<http://jsonlint.com/>

## 17. APPENDIX 5 - BACKGROUND INFORMATION

### 17.1 MODBUS BACKGROUND

Modbus is a serial communications protocol published by Modicon in 1979 for use with its programmable logic controllers (PLCs). The Modbus standard is currently managed by The Modbus Organization. The standard is available for free download from <http://modbus.org/specs.php>. Download the "Modbus Serial Line Protocol and Implementation Guide" (Modbus\_over\_serial\_line\_V1\_02.pdf) and the "MODBUS Protocol Specification" (Modbus\_Application\_Protocol\_V1\_1b3.pdf)

Modbus is a simple request-response protocol, in which a master device sends a message asking for a particular slave device to return a number of registers, each containing information collected or derived from the devices and sensors connected to the slave.

It can be implemented over Serial (Modbus ASCII / Modbus RTU) or Ethernet (Modbus TCP/IP). "Modbus RTU" (Remote Terminal Unit) uses raw binary encoding, whilst "Modbus ASCII" uses ASCII characters (7 bits). The SmartSwarm 351 currently supports eavesdropping on serial Modbus RTU networks only.

There are a number of ways to specify the addresses of registers in the slave device. SmartSwarm 351 uses the convention of a base register type (Coil, Status, Input or Holding) and an offset, starting from zero, within that register type. Hence, a register defined within SmartSwarm 351 as 'Holding Register 5) might appear in other manufacturers data as register 40,006, or 400,006.

SmartSwarm 351 decodes information exchanged using the following Modbus commands:

Function Code	Description
01	Read Coil Status
02	Read Input Status
03	Read Holding registers
04	Read Input registers
05	Force Single Coil*
06	Preset Single register*
15	Force multiple Coils*
16	Preset multiple registers*
22	Mask Write 4X register*
23	Read/Write 4X registers*

Table 51. Supported Modbus commands

\* For output command types, the unit interprets these as inputs to the IoT enrichment process. It does not support output of data from the IoT system.

The following types of data recovered in Modbus registers can be interpreted and decoded by SmartSwarm 351:

Data Type
Boolean
Multi-bit Encoded Boolean (e.g. 2 bits provide 4 separate states for one point)
16 bit Packed Boolean (i.e. 16 discrete Booleans held in a single 16 bit register)
16 bit Integer (signed/ unsigned)
16 bit Counter
32 bit Integer (signed/unsigned) (single 32 bit or 2x16 bit registers)
32 bit Counter (single 32 bit or 2x 16 bit registers)
32 bit Float (single 32 bit or 2x 16 bit registers)
32 bit Packed Boolean (single 32 bit or 2x 16 bit registers)
Multi-register text

**Table 52. Supported Data Types**

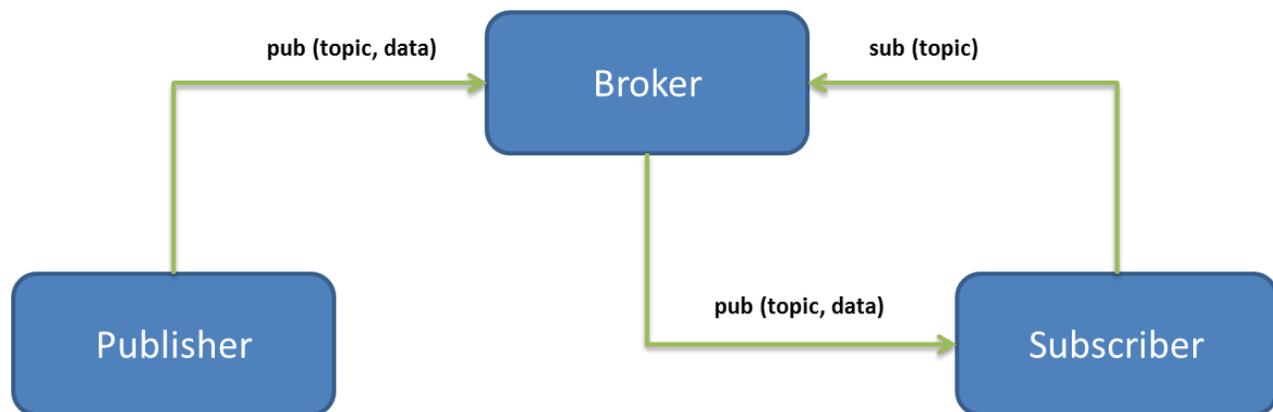
SmartSwarm also supports big and little endian formats with both byte swap and, in the case of 32 bit values being transmitted in two consecutive 16 bit registers, register swap capabilities.

## 17.2 MQTT BACKGROUND

MQTT (Message Queuing Telemetry Transport) is a transport protocol originally developed in 2001 by IBM and Arcom specifically to address the need for reliable, pushed data delivery for M2M systems over unreliable networks. It has subsequently been released as open source, ratified by OASIS as an open standard for IoT protocols, and most recently has been standardized as ISO/IEC 20922. It is a relatively simple solution designed to facilitate the efficient and scalable transfer of data from multiple (potentially different) devices and provide the information to several diverse applications, while providing 24/7 reliability.

MQTT communications are based upon a publish/subscribe methodology. Data sources 'publish' data payloads on a particular data topic. Data consumers 'subscribe' to topics of interest and receive all packets published on that topic in near real-time. Devices and applications can be both publishers of, and subscribers to, data topics. MQTT is a very lightweight protocol and is therefore very useful in applications where processor resources, memory or communications bandwidth are limited.

### The publish/subscribe communication model



MQTT uses an intermediate middleware server to keep track of all of the subscriptions in use from connected devices. When it receives a message the server (also referred to as a 'broker') analyses the 'topic' information contained in the message, and simply forwards the message to any and all applications with a matching topic subscription. While the above diagram shows a transaction from a single source to a single subscriber application, it should be realized that multiple subscribers can access the same topics concurrently, enabling one-to-many communication models. It should also be remembered that any device can be both a publisher of and a subscriber to data topics, enabling bidirectional communications.

### Topic Naming

The topic naming mechanism within MQTT uses a hierarchical subject format and can contain any of the characters found in the Unicode character set. (e.g. company/city/building/room{data payload}). Topic (and subtopic) names may be selected on an individual project basis to suit the needs of the application, but can be extended and modified at any time as the application evolves. Much of the power of MQTT comes from the ability to include wildcards within the subscription definition, either for single or multiple levels within the topic name hierarchy.

- A '#' character represents a complete sub-tree of the hierarchy and thus must be the last character in a subscription topic string, such as SENSOR/#. This will match any topic starting with SENSOR/, such as SENSOR/1/TEMP and SENSOR/2/HUMIDITY.
- A '+' character represents a single level of the hierarchy and is used between delimiters. For example, SENSOR+/TEMP will match SENSOR/1/TEMP and SENSOR/2/TEMP.

With a carefully designed topic space it becomes possible to implement very powerful searching, filtering and combination of data simply by manipulation of the wildcards used within subscriptions.

**There are a few rules to remember when designing a topic space schema:**

- Topic names are case-sensitive. For example, "CITY", "City" and "city" are all recognized as different topic names.
- Topic names can include spaces, which are treated just like any other character. "building A" and "building B" are both valid constructs representing different entities.
- While it is not recommended, a topic level may contain a null string. For example, "company//building" is a three level topic name whose middle level is empty.
- It is recommended that topic names do not include the null character (Unicode \x0000).
- There is no effective limit to length of the overall topic name string. (But it should be remembered that the topic is transmitted and so overly long topics may result in high data charges if using, for example, cellular communications).
- There is no limit to the levels of depth (number of slash-separated strings) in a topic tree.
- There is no limit to the length of any particular level name in the tree (but again, this will impact data usage).
- There may be any number of "root" nodes (that is, any number of topic trees).

Topic	Description
MyCo/Galway/building A/Warehouse	Data relating to the warehouse in building A located in Galway for my company
MyCo/Galway+/Warehouse	Data for warehouse facilities in all MyCo buildings in Galway
MyCo/Galway/building A/+	Data from all rooms in building A operated by MyCo in Galway
MyCo/Galway/#	All data from all buildings operated by MyCo located in Galway
MyCo/#	All data from all MyCo buildings

**Table 53. Examples for subscribing to different topics in a hierarchical name space**

As can be seen from the above table, a well-designed topic space will segregate data into subject trees which simplify the gathering of data in the cloud.

**System Efficiency and Reliability**

MQTT was conceived for use in 24/7, mission critical applications. As such, MQTT includes concepts such as 'Quality of Service', which allow users to control how reliably messages are received by subscribers. Fire & forget, deliver at least once, or deliver once and once only, each with different levels of handshake within the underlying messaging system.

Also included is a 'last will & testament' message. This is a special category of message which, when published, is not immediately forwarded to all subscribers. Instead, it is held by the middleware server on behalf of the publishing device, and is republished should the originating unit disappear from the network in an unexpected

manner. This allows applications to understand the difference between a unit which is not publishing data because nothing has happened, and one which is not publishing data because it has failed – a critical requirement for push based messaging systems.

**MQTT Summary**

MQTT is a widely adopted, lightweight open protocol which provides a transport layer in IoT architectures. Using Publish & Subscribe methodology, it decouples the users of data from the producers of that data. This allows extremely flexible and scalable data exchange architectures systems to be constructed, which use semantic principles to move the users' focus away from how to move the data between systems, and towards how the available data may be combined to produce business benefit.

18. APPENDIX 6 – DASHBOARDS

Having enriched our network and added rules and topics to publish the data we are interested in, we now need a way to display this data. There are several ways to visualize the published data.

- Use an MQTT client such as [MQTT-Spy](#)
- Browser extension such as [MQTTLens](#)
- MQTT [Treeview](#)

All of these work by subscribing to the MQTT Server and topic.

We can also visualize the published data using dashboards such as [Freeboard](#) or [Node-RED UI](#)

Please note that Advantech B+B SmartWorx does not offer Dashboarding software: This section is for example purposes only.

18.1 NODE-RED

Using Node-RED, a dashboard can be quickly created to display your MQTT data.

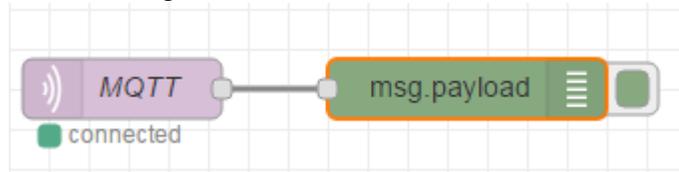
In this example we will set up two rules publishing to two different topics:

Address	Type	Name	Units	Event	Payload	QOS	Retain	MQTT Topic	Default Topic	
5	HR	Speed	rpm	Read	Default	Exactly Once	<input type="checkbox"/>	Oranmore/Pump/P1/SP	<input checked="" type="checkbox"/>	- +
6	HR	Power	kWh	Read	Default	Exactly Once	<input type="checkbox"/>	Oranmore/Pump/P1/PC	<input checked="" type="checkbox"/>	- +

This will result in an MQTT payload in the following format

```
{
  "id": "Oranmore/Pump/P1/SP",
  "model": {
    "meta": {
      "address": {
        "DEVID": "70000000",
        "PORTID": 1,
        "SLAVEID": 5,
        "SWMID": "0"
      },
      "description": "Pump",
      "installation_date": "29/03/2016",
      "location": "Oranmore",
      "manufacturer": "",
      "name": "P1",
      "product_code": "",
      "value_byte_order": "SNo",
      "version": ""
    },
    "state": {
      "HR": [
        {
          "address": 5,
          "alias": "",
          "at": "2016-06-11T21:45:55.183Z",
          "name": "Speed",
          "new_read": true,
          "new_value": true,
          "num_value": 300,
          "published_on": "READ",
          "state": "VALIDATED",
          "value_from": "RESPONSE"
        }
      ]
    }
  },
  "type": "ModbusSlave"
}
```

- Add an MQTT node and a Debug node to the Node-RED flow and connect the 2 nodes



- Configure the MQTT node by entering the MQTT broker IP Address and the topic we want to subscribe to. In this example, the topic is Oranmore/Pump/P1.

**Edit mqtt in node**

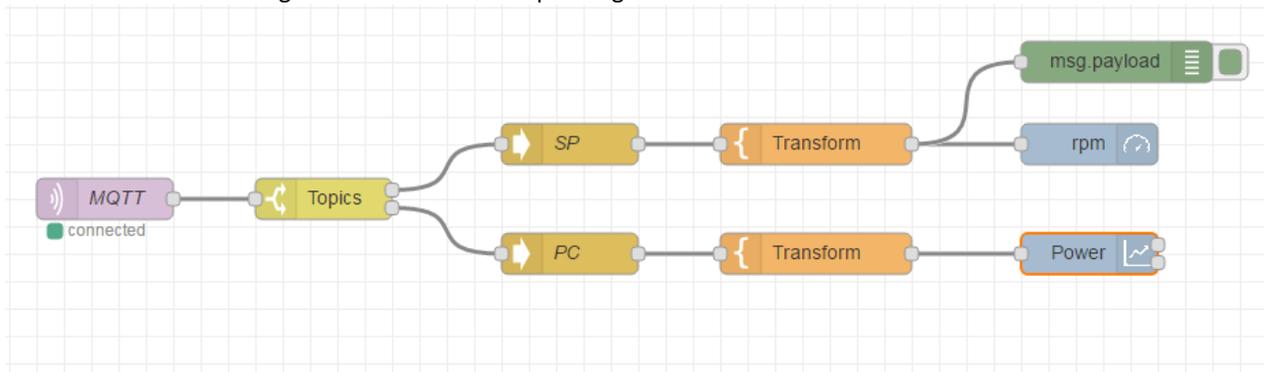
Server

Topic

Name

- Once you have configured the MQTT node, click **Deploy**. Check that the MQTT payload is being received by clicking on the debug node. The output will be printed in the debug tab.

- Add the following nodes and connect as per diagram below



Topics is a switch node which gives multiple outputs based on rules. Add two rules as per the diagram below and click **OK**. In this example two different topics are being published, so we add an equal rule for each topic. This means that if Rule 1 is matched the payload will be output on output 1. Matches for Rule 2 will be output on output 2.

Edit switch node

Name

Property

- ==  → 1
- ==  → 2

SP and PC are JSON nodes which transform the input into a JSON object.

Transform is a Template node which allows us to select the payload property as an input to another node. In this case, num\_value is the payload property that we want. The template will depend on the Register type and, in the case of multiple registers, the index of the register within the register array. In this case, the desired register is HR and it is the register at index 0.

### Edit template node

 Name  

 Set property

 Template Syntax Highlight:

```
1 {{payload.model.state.HR.0.num_value}}
```

 Format

---

rpm is a Gauge node on which we will display the Oranmore/Pump/P1/SP topic.

**Edit ui\_gauge node**

Tab  ✎

Name

Group  Order

Template

Min

Max

---

Click the pencil icon to create a new tab called **Example**. This will be the name of the menu on our page which we will click to view the dashboard. Fill in the other fields as per the diagram below and click **Ok**.

Field	Description
Tab	Menu under which the control will be displayed
Name	Title of the control on the page
Group	Controls can be grouped on the page in a vertical row
Order	The order the control is displayed in the vertical row
Template	This will always be {{value}} which is the input value from the connected node
Min	This is the minimum value specified during enrichment for the register
Max	This is the maximum value specified during enrichment for the register

Table 54. Node Red fields for Gauge node

Edit ui\_gauge node

Tab  ▼ ✎  
Name   
Group  ↕ Order   
Template   
◀ Min   
▶ Max

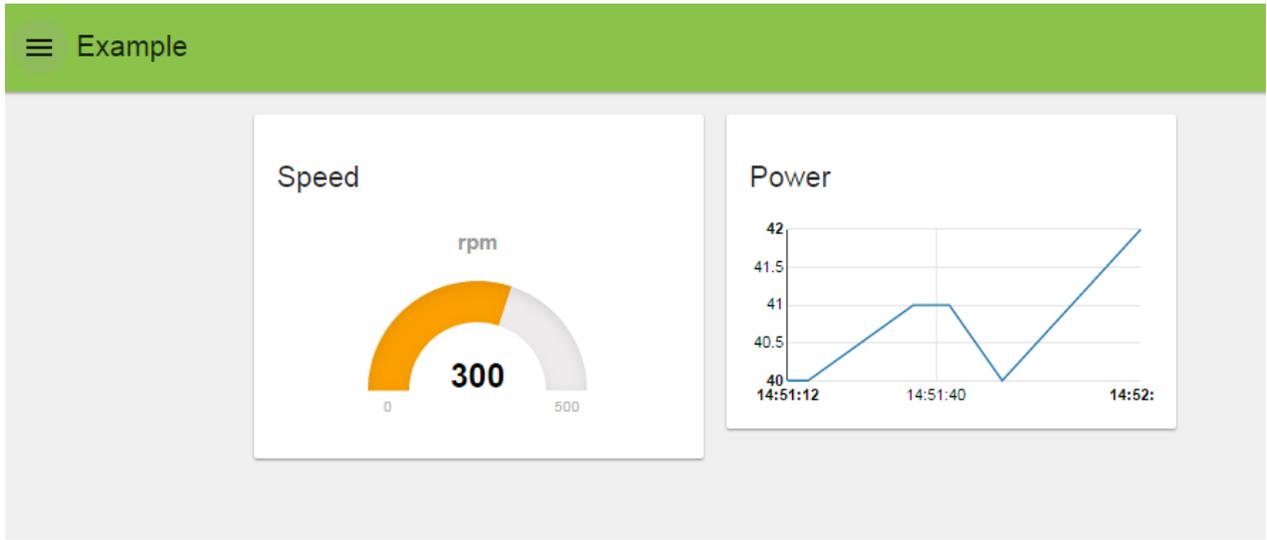
Ok

Cancel

Power is a chart node on which we will display the Oranmore/Pump/P1/PC topic. Fill in the other fields as per the diagram below and click **Ok**. In this case we choose the Example Tab, so it is available under the same menu as the gauge control.

Field	Description
Tab	Menu under which the control will be displayed
Name	Title of the control on the page
Group	Controls can be grouped on the page in a vertical row
Order	The order the control is displayed in the vertical row
Old After	Period of time after which oldest data will be removed from the chart
No Data	Message to be displayed when no data has been received
Interpolate	Type of graph to be displayed

Table 55. Node Red fields for Chart node



As the data changes it will automatically update the controls on the page.

Edit ui\_chart node

Ok Cancel

All the nodes are now configured and connected, and are deployed by clicking **Deploy**. The message **Successfully deployed** will be displayed. Open a new tab in the browser and add the node-RED server address/ui to the address bar. The dashboard should now be displayed.

**ADVANTECH B+B SMARTWORX TECHNICAL SUPPORT**

**Phone:** 1-800-346-3119  
(Monday - Friday, 7 a.m. to 5:30 p.m. CST)  
**Fax:** 815-433-5109  
**Email:** [support@advantech-bb.com](mailto:support@advantech-bb.com)  
**Web:** [www.advantech-bb.com](http://www.advantech-bb.com)