User Manual

WISE-4250 Series

ADVANTECH

Enabling an Intelligent Planet

Copyright

The documentation and the software included with this product are copyrighted 2024 by Advantech Co., Ltd. All rights are reserved. Advantech Co., Ltd. reserves the right to make improvements in the products described in this manual at any time without notice. No part of this manual may be reproduced, copied, translated, or transmitted in any form or by any means without the prior written permission of Advantech Co., Ltd. The information provided in this manual is intended to be accurate and reliable. However, Advantech Co., Ltd. assumes no responsibility for its use, nor for any infringements of the rights of third parties that may result from its use.

Acknowledgments

Intel and Pentium are trademarks of Intel Corporation.

Microsoft Windows and MS-DOS are registered trademarks of Microsoft Corp.

All other product names or trademarks are properties of their respective owners.

Product Warranty (2 years)

Advantech warrants the original purchaser that each of its products will be free from defects in materials and workmanship for two years from the date of purchase.

This warranty does not apply to any products that have been repaired or altered by persons other than repair personnel authorized by Advantech, or products that have been subject to misuse, abuse, accident, or improper installation. Advantech assumes no liability under the terms of this warranty as a consequence of such events.

Because of Advantech's high quality-control standards and rigorous testing, most customers never need to use our repair service. If an Advantech product is defective, it will be repaired or replaced free of charge during the warranty period. For out-of-warranty repairs, customers will be billed according to the cost of replacement materials, service time, and freight. Please consult your dealer for more details.

If you believe your product to be defective, follow the steps outlined below.

1. Collect all the information about the problem encountered. (For example, CPU speed, Advantech products used, other hardware and software used, etc.) Note anything abnormal and list any onscreen messages displayed when the problem occurs.
2. Call your dealer and describe the problem. Please have your manual, product, and any helpful information readily available.
3. If your product is diagnosed as defective, obtain a return merchandise authorization (RMA) number from your dealer. This allows us to process your return more quickly.
4. Carefully pack the defective product, a completed Repair and Replacement Order Card, and a proof of purchase date (such as a photocopy of your sales receipt) into a shippable container. Products returned without a proof of purchase date are not eligible for warranty service.
5. Write the RMA number clearly on the outside of the package and ship the package prepaid to your dealer.
6.

Part No. 2001425011 Edition 2

Printed in Taiwan December 2024

Declaration of Conformity

CE

This product has passed the CE test for environmental specifications when shielded cables are used for external wiring. We recommend the use of shielded cables. This type of cable is available from Advantech. Please contact your local supplier for ordering information.

Test conditions for passing also include the equipment being operated within an industrial enclosure. In order to protect the product from damage caused by electro-static discharge (ESD) and EMI leakage, we strongly recommend the use of CE-compliant industrial enclosure products.

Technical Support and Assistance

1. Visit the Advantech website at www.advantech.com/support to obtain the latest product information.
2. Contact your distributor, sales representative, or Advantech's customer service center for technical support if you need additional assistance. Please have the following information ready before calling:
   - Product name and serial number
   - Description of your peripheral attachments
   - Description of your software (operating system, version, application software, etc.)
   - A complete description of the problem
   - The exact wording of any error messages

Warnings, Cautions and Notes

Warning! *Warnings indicate conditions that if not observed can cause personal injury!*

Caution! *Cautions are included to help prevent hardware damage and data losses. For example,*

"Batteries are at risk of exploding if incorrectly installed. Do not attempt to recharge, force open, or heat the battery. Replace the battery only with the same or equivalent type as recommended by the manufacturer. Discard used batteries according to the manufacturer's instructions."

Note! *Notes provide additional optional information.*

Safety Instructions

1. Read these safety instructions carefully.
2. Retain this user manual for future reference.
3. Disconnect the equipment from all power outlets before cleaning. Use only a damp cloth for cleaning. Do not use liquid or spray detergents.
4. For pluggable equipment, the power outlet socket must be located near the equipment and easily accessible.
5. Protect the equipment from humidity.
6. Place the equipment on a reliable surface during installation. Dropping or letting the equipment fall may cause damage.
7. The openings on the enclosure are for air convection. Protect the equipment from overheating. Do not cover the openings.
8. Ensure that the voltage of the power source is correct before connecting the equipment to a power outlet.
9. Position the power cord away from high-traffic areas. Do not place anything over the power cord.
10. All cautions and warnings on the equipment should be noted.
11. If the equipment is not used for a long time, disconnect it from the power source to avoid damage from transient overvoltage.
12. Never pour liquid into an opening. This may cause fire or electrical shock.
13. Never open the equipment. For safety reasons, the equipment should be opened only by qualified service personnel.
14. If any of the following occurs, have the equipment checked by service personnel:
    - The power cord or plug is damaged.
    - Liquid has penetrated the equipment.
    - The equipment has been exposed to moisture.
    - The equipment is malfunctioning, or does not operate according to the user manual.
    - The equipment has been dropped and damaged.
    - The equipment shows obvious signs of breakage.
15. Do not leave the equipment in an environment with a storage temperature of below -20 °C (-4 °F) or above 60 °C (140 °F) as this may damage the components. The equipment should be kept in a controlled environment.
16. CAUTION: Batteries are at risk of exploding if incorrectly replaced. Replace only with the same or equivalent type as recommended by the manufacturer. Discard used batteries according to the manufacturer’s instructions.
17. In accordance with IEC 704-1:1982 specifications, the sound pressure level at the operator's position does not exceed 70 dB (A).

DISCLAIMER: These instructions are provided according to IEC 704-1 standards. Advantech disclaims all responsibility for the accuracy of any statements contained herein.

Safety Precaution - Static Electricity

Follow these simple precautions to protect yourself from harm and the products from damage.

- To avoid electrical shock, always disconnect the power from the PC chassis before manual handling. Do not touch any components on the CPU card or other cards while the PC is powered on.
- Disconnect the power before making any configuration changes. A sudden rush of power after connecting a jumper or installing a card may damage sensitive electronic components.

NCC 警語

第十二條 經型式認證合格之低功率射頻電機，非經許可，公司、商號或使用者均不得擅自變更頻率、加大功率或變更原設計之特性及功能。

第十四條 低功率射頻電機之使用不得影響飛航安全及干擾合法通信；經發現有干擾現象時，應立即停用，並改善至無干擾時方得繼續使用。前項合法通信，指依電信法規定作業之無線電通信。低功率射頻電機須忍受合法通信或工業、科學及醫療用電波輻射性電機設備之干擾。

Federal Communication Commission Interference Statement

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

FCC Caution: Any changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate this equipment.

This transmitter must not be co-located or operating in conjunction with any other antenna or transmitter.

FOR MOBILE DEVICE USAGE (>20cm/low power)

Radiation Exposure Statement

This equipment complies with FCC radiation exposure limits set forth for an uncontrolled environment. This equipment should be installed and operated with minimum distance 20cm between the radiator & your body.

FOR COUNTRY CODE SELECTION USAGE (WLAN DEVICES)

Note: The country code selection is for non-US model only and is not available to all US model. Per FCC regulation, all WiFi product marketed in US must fixed to US operation channels only.

Industry Canada Statement:

This device complies with ISED's licence-exempt RSSs. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

Le présent appareil est conforme aux CNR d' ISED applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes : (1) le dispositif ne doit pas produire de brouillage préjudiciable, et (2) ce dispositif doit accepter tout brouillage reçu, y compris un brouillage susceptible de provoquer un fonctionnement indésirable.

FOR MOBILE DEVICE USAGE (>20cm/low power)

Radiation Exposure Statement

This equipment complies with ISED radiation exposure limits set forth for an uncontrolled environment. This equipment should be installed and operated with minimum distance 20cm between the radiator & your body.

Déclaration d'exposition aux radiations

Cet équipement est conforme aux limites d'exposition aux rayonnements ISED établies pour un environnement non contrôlé. Cet équipement doit être installé et utilisé avec un minimum de 20 cm de distance entre la source de rayonnement et votre corps.

Contents

Chapter 1

Product Overview

1.1 Series Family and Specifications

Table 1.1: Series Family and Specifications

Function	Model	Description
Wireless IO/Sensor node	WISE-4250	2.4/5GHz Wi-Fi IoT Wireless I/O & Sensor Node

1.2 Mechanical Design and Dimensions

1.2.1 WISE-4250 Series Dimension

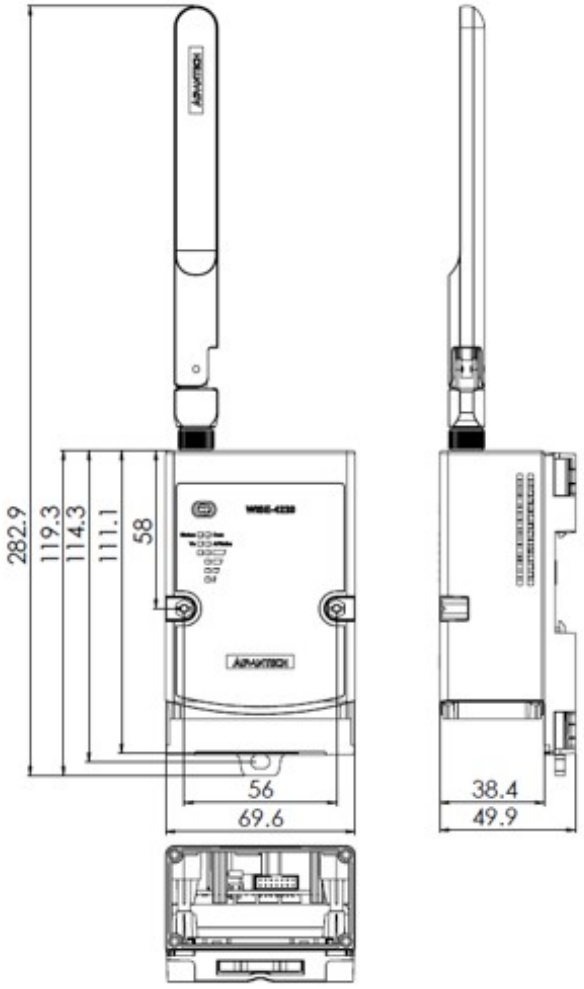


Figure 1.1 Mechanical Dimension Front, Side and Bottom

1.3 Switch

Open the housing behind the WISE-4250 module, you can see a switch inside. The battery socket is no function for WISE-4250. It is reserved for WISE-4210 series.

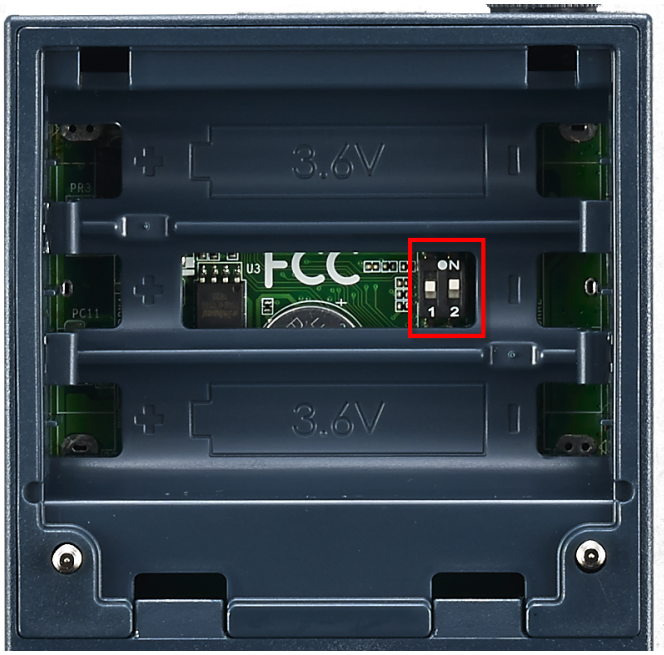


Table 1.2: Switch

Switch	Description	Position	ON (Default)	OFF
SW1	Operation Mode	P1	Normal Mode	Initial Mode
		P2	N/A	N/A

Note! *After the position 1 of SW1 been changed, users need to power on the module again to apply the operation mode.*

1.4 LED Definition

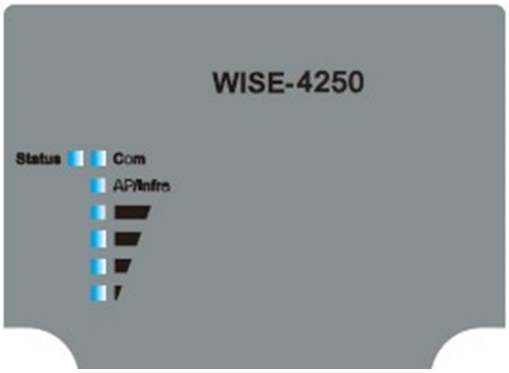


Table 1.3: LED Definition

LED	Color	Indication	Behavior
Status	Green	Blink	2Hz: Wait for connection 0.5Hz: Network Connected
		ON 30 Sec	When enable LOCATE function
Com	Yellow	Blink	When TX/RX data in transmission
AP/Infra	Green	ON	Limited AP Mode
		OFF	Station Mode
Signal Strength	Green	ON *4	Full Signal (Blink: RF module fail)
		ON *3	Good Signal (Blink: I/O module fail)
		ON *2	Okay Signal
		ON *1	Poor Signal
		All OFF	No Signal / AP Module

1.5 Certification and Safety Standard

Electromagnetic Compatibility (EMC)

- CE
  - EN 55011:2016 +A1:2017+A11:2020 (Group I, Class B)
  - EN 55032:2015 +A11:2020, Class A
  - EN 55035:2017 +A11:2020
  - EN 61000-4-2:2009/2008 ED.2.0
  - EN 61000-4-3:2020/2020 ED.4.0
  - EN 61000-4-4:2012/2012 ED.3.0
  - EN 61000-4-5:2014+A1:2017/2017 ED3.1
  - EN 61000-4-6:2014+AC:2015/2013 ED.4.0
  - EN 61000-4-8:2010/2009 ED.2.0
  - EN 61000-4-11:2020/2020 ED.3.0
  - EN 61000-6-2:2019
  - EN 61000-6-4:2019
  - EN 301 489-1 V2.2.3 (2019-11)
  - EN 301 489-3 V2.3.2 (2023-01)
  - EN 301 489-17 V3.2.4 (2020-09)

- FCC
  - 47 CFR FCC Part 15, Subpart B, Class A
  - ICES-003:2020 Issue 7, Class A
  - ICES-Gen:2018 Issue 1+A1:2021
  - ANSI C63.4:2014
- CISPR
  - AZ/NZS CISPR 32:2015+AMD1:2020, Class A
  - CISPR 32:2015+AMD1:2019 ED.2.0, Class A
  - CISPR 32:2019 ED.2.1, Class A
- Safety
  - IEC 62368-1:2014(Second Edition)+A11:2017

Wireless Certification

- FCC Contains Wi-Fi Module AW-CM358 ID: TLZ-CM358SM
- IC Contains Wi-Fi Module AW-CM358 ID:6100A-CM358SM
- TELEC Contains Wi-Fi Module AW-CM358 ID: 020-210069
- RED
  - EN 300 328 V2.2.2 (2016-11)
  - EN 63211:2008

1.6 Package Information

WISE-4250-A

- WISE-4250 module with bundle antenna and terminal connector x1
- Mounting bracket x1
- Quick startup manual with China RoHS declare

Chapter 2

General Specification

2.1 General Specification

2.1.1 WLAN Interface

- **Network Modes**
  - Infrastructure/Station (Wireless Client)
  - Limited AP (Wireless Server, Allow one client for local connection to do configuration)
- **Standard Conformance**
  - 2.4GHz
    - 802.11b
    - 802.11g
    - 802.11n
  - 5 GHz
    - 802.11a
    - 802.11n
    - 802.11ac
- **Modulation**
  - 802.11b: CCK (11, 5.5Mbps), DQPSK (2Mbps),
  - BPSK(1Mbps)
  - 802.11a/g/n/ac: OFDM
- **Frequency Range**
  - 2.4 GHz: 2.412-2.472 GHz
  - 5GHz: 5.15-5.35 GHz & 5.47-5.825 GHz
- **Channels**
  - 2.4 GHz: ch1 ~ ch13
  - 5GHz: ch36, ch40, ch44, ch48, ch52, ch56, ch60, ch64, ch100, ch104, ch108,ch112, ch116, ch120, ch124, ch128, ch132, ch136, ch140, ch149, ch153, ch157, ch161, ch165
- **Transmit Power**
  - 2.4 GHz
    - 802.11b: 16.0 dBm ±2dBm
    - 802.11g: 14.0 dBm ±2dBm
    - 802.11n: 12.0 dBm ±2dBm
  - 5 GHz
    - 802.11a: 13.0 dBm ±2dBm
    - 802.11n: 10.0 dBm ±2dBm
    - 802.11ac: 8.0 dBm ±2dBm
- **Receiver Sensitivity**
  - 2.4 GHz
    - 802.11b: -84dBm (Max)
    - 802.11g: -70dBm (Max)
    - 802.11n: -64dBm (Max)
  - 5 GHz
    - 802.11a: -68dBm (Max)
    - 802.11n: -60dBm (Max)
    - 802.11ac: -52dBm (Max)
- **Wireless Security**
  - WPA2 Personal and

- Enterprise, WPA3 personal
- Antenna
- Connector: RP-SMA
- Gain (Peak): 2.4G 3.64 dBi / 5G 5.65 dBi (Max)

2.1.2 General

- **I/O Connector:** 3.5mm spacing plug-in screw terminal block
- **Power Connector:** 3.5mm spacing plug-in screw terminal block
- **Watchdog Timer**
- **RTC Accuracy:** ±1 min/month
- **Enclosure:** PC
- **Mounting:** DIN 35 rail, wall, pole and stack
- **Dimensions (W x H x D):** 69 x 38 x 102 mm
- **Operation Temperature:** -25~70 °C (-13~158 °F)
- **Storage Temperature:** -40~85 °C (-40~185 °F)
- **Operating Humidity:** 10~ 85% RH (non-condensing)
- **Storage Humidity:** 0~60% RH (non-condensing)

Note! *Equipment will operate below 30% humidity. However, static electricity problems occur much more frequently at lower humidity levels. Make sure you take adequate precautions when you touch the equipment. Consider using ground straps, anti-static floor coverings, etc. if you use the equipment in low humidity environments.*

Note! *Measuring temperature and humidity will depend on sensor type. Whether the device is measuring temperature or humidity depends on the settings of the sensors.*

2.1.3 Power

- **Power Input Voltage:** External Power: 10 ~ 50 V_{DC}
- **Power Consumption:** 1.6 W @ 24 V_{DC}

2.1.4 Software

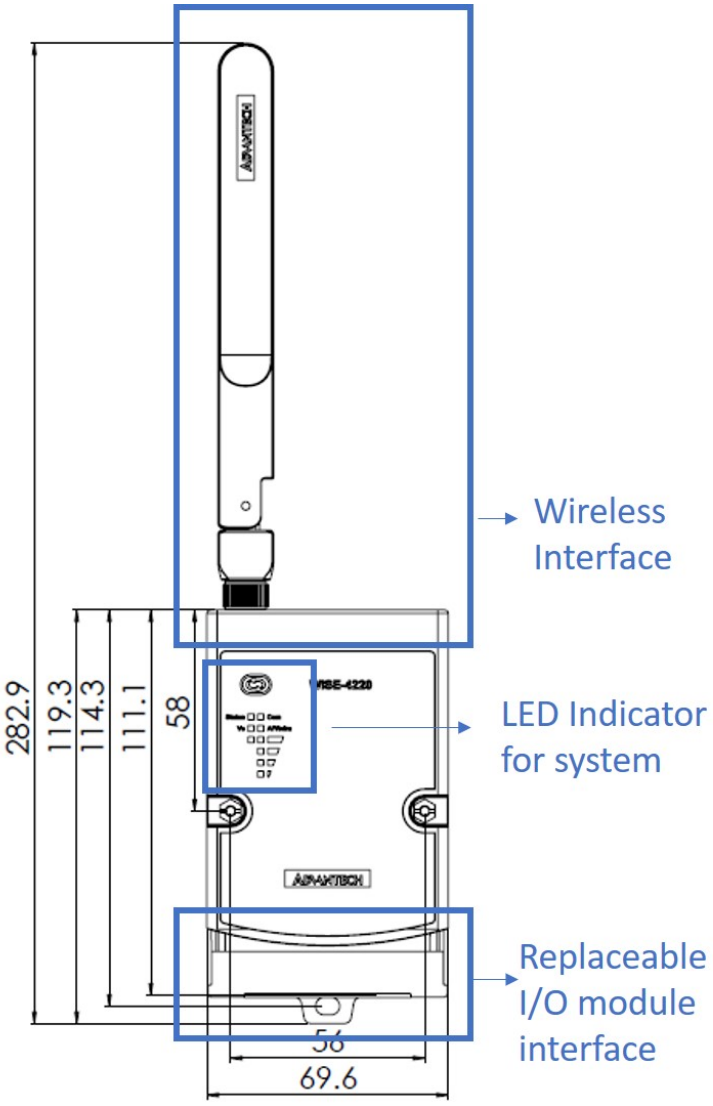
- **Configuration Interface:** Web Interface, Windows Utility
- **Utility:** WISE Studio Utility
- **Driver:** ADAM .NET Class Library
- **Supported Protocols:** Modbus/TCP, MQTT, TCP/IP, SNMP v2, SNTP, UDP, DHCP, HTTP, and HTTPS
- Supports RESTful Web API in JSON format
- Supports Web Server in HTML5 with JavaScript & CSS3
- Support wireless P2P (Peer to Peer) with AES-128 encryption and UDP protocol
- Supports Cloud: Dropbox, WebAccess, Azure, AWS, Line Notify, etc.
- Supports MQTT data recovery function
- Supports the smart roaming function and 802.11r standard.
- Supports Flexible Modbus Address
- Supports Data Log 10000+ samples with SNTP/RTC sync time stamp Function
- Supports System Configuration Backup and User Access Control

Chapter 3

Hardware Installations

3.1 Interface Introduction

Detailed description of the WISE-4250 interface is shown below.



3.2 Mounting

WISE-4250 modules are designed as compact units and are allowed to be installed in the field site under the following methods.

3.2.1 DIN-Rail Mounting

The WISE-4250 module can also be fixed to the cabinet by using mounting rails. You need to assemble the DIN rail adapter to WISE-4250 module with flathead screw driver as below. When the module is mounted on a rail, you may also consider using end brackets at each end of the rail to keep the module from sliding horizontally along the rail.

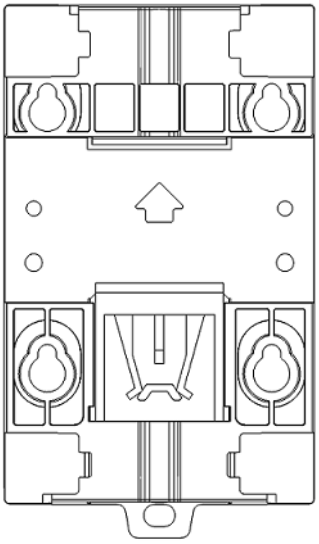


Figure 3.1 DIN Mounting Kit

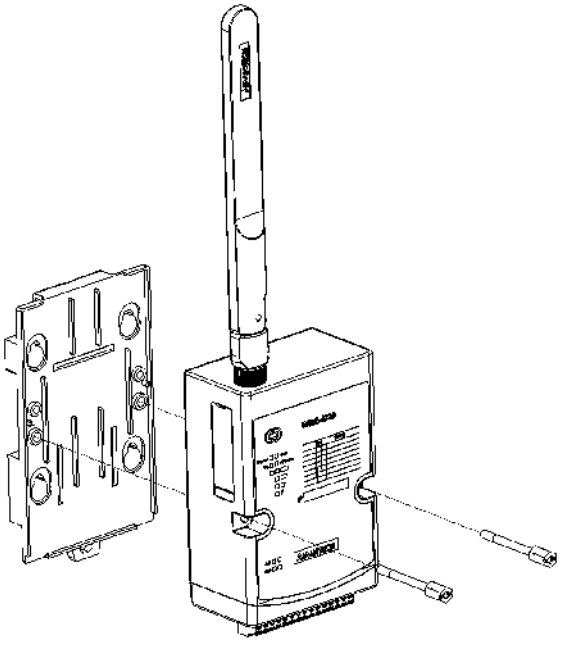


Figure 3.2 DIN Mounting Install

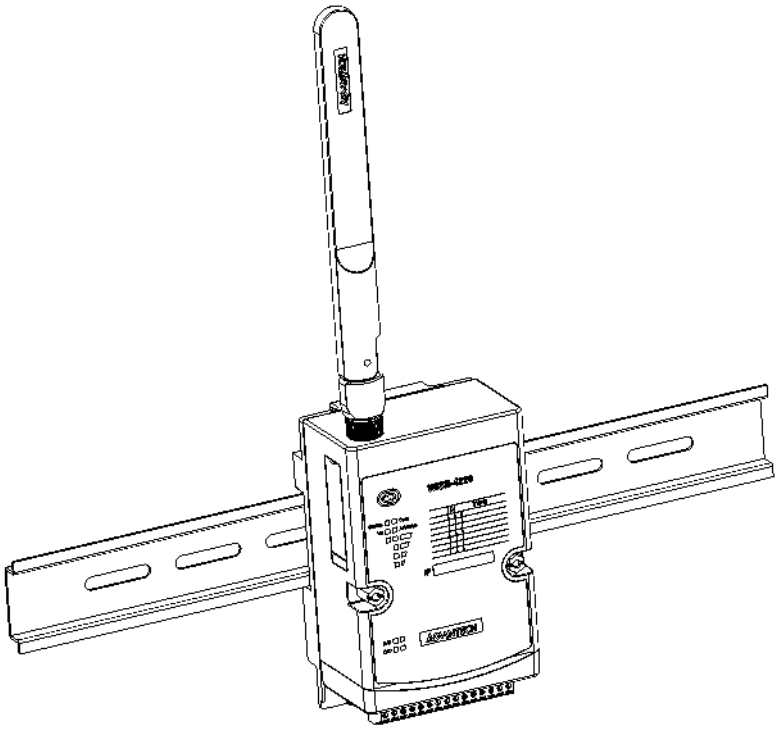


Figure 3.3 DIN Mounting Front

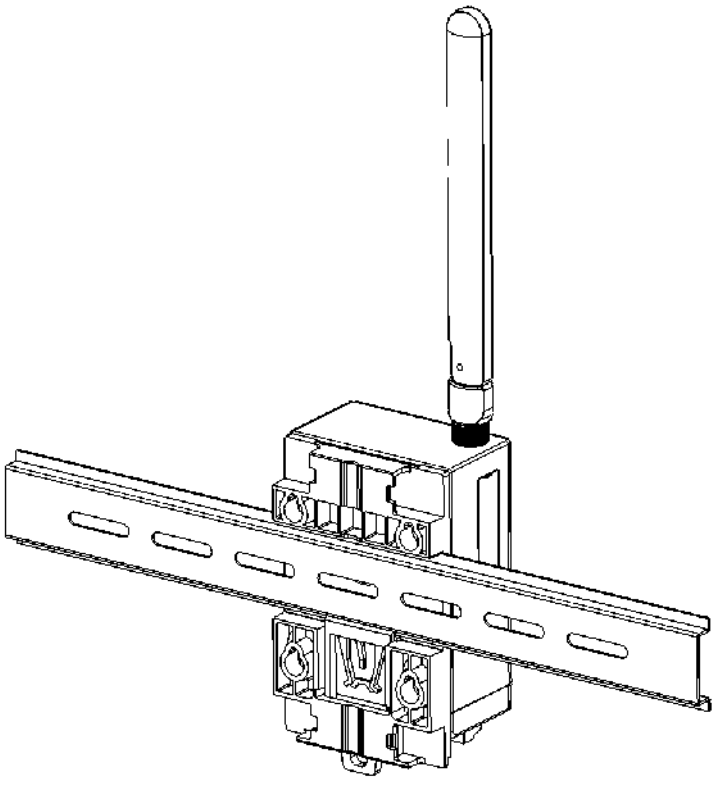


Figure 3.4 DIN Mounting Back

3.2.2 Wall Mounting

Each WISE-4250 module is packed with a plastic wall mounting bracket. Users can refer to the bracket dimensions and assembly steps to configure an optimal placement in a wall, panel, or cabinet.

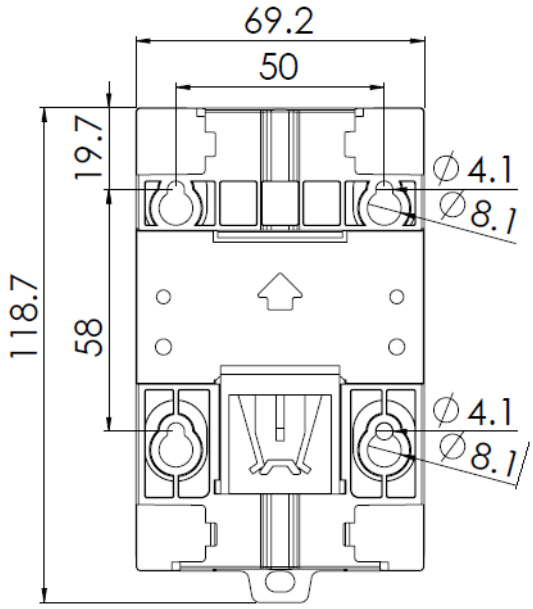


Figure 3.5 Wall Mounting Kit Dimension

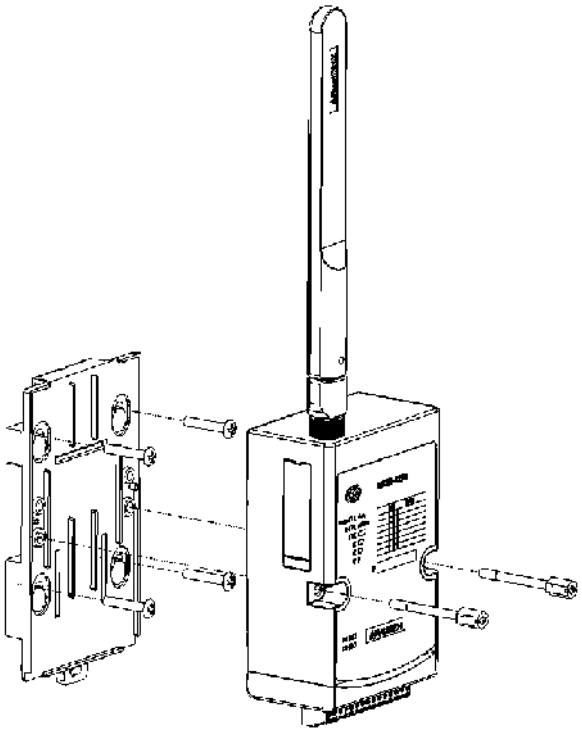


Figure 3.6 Wall Mounting Install 1

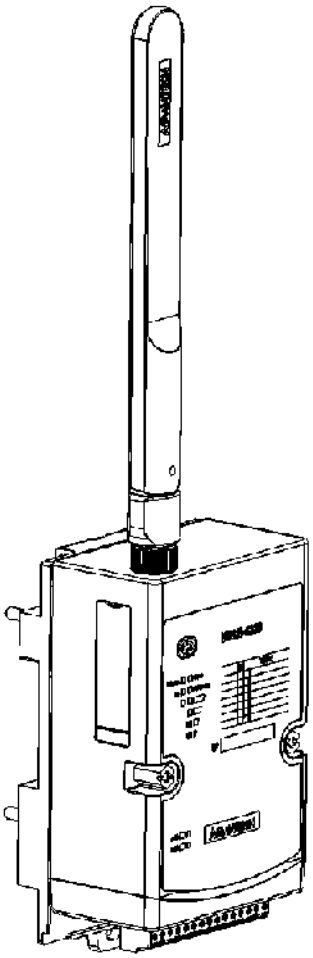


Figure 3.7 Wall Mounting Install 2

3.2.3 Pole Mounting

Put the pole mounting ring through the middle hole of it. Note that you should unlock the pole mounting ring with a screw driver before putting it through the device. Then mount the WISE-4250 module steadily to the pole by locking the pole mounting ring tightly.

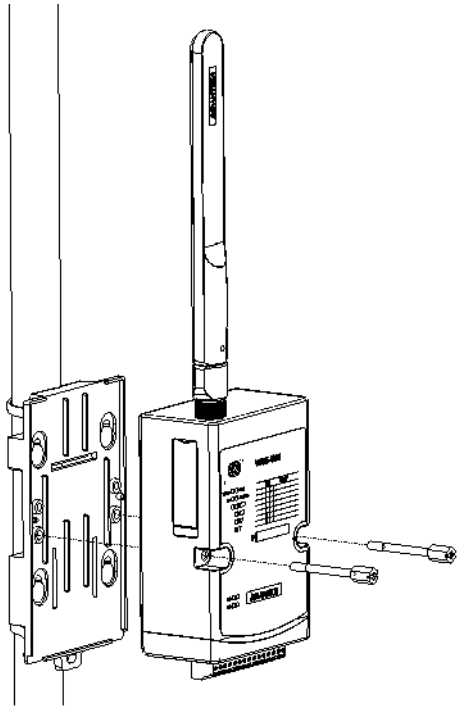


Figure 3.8 Polar Mounting Front

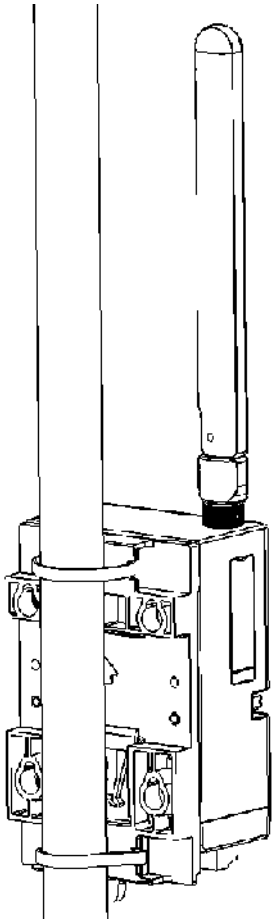


Figure 3.9 Polar Mounting Back

3.3 Wiring & Connections

This section introduces basic information on wiring the power supply and I/O units.

3.3.1 Power Supply Wiring

The system of WISE-4250 is designed for a standard industrial unregulated 24 V_{DC} power supply. For further application, it can also accept +10 to +50 V_{DC} of power input, 200mV peak to peak of power ripple, and the immediate ripple voltage should be maintained between +10 and +50 V_{DC}. Screw terminals +Vs and -Vs are for power supply wiring.

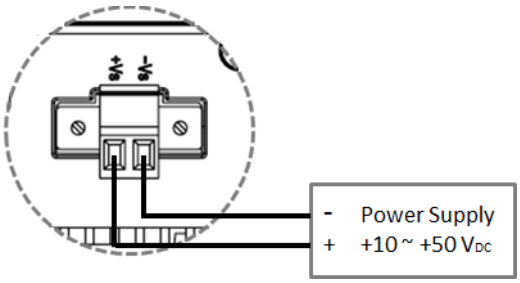


Figure 3.10 Power Wiring

3.3.2 I/O Units

The system uses a plug-in screw terminal block for the interface between I/O modules and field devices. The following information must be considered when connecting electrical devices to I/O modules.

- The terminal block accepts wires from 0.5 mm to 2.5 mm.
- Always use a continuous length of wire. Do not combine wires.
- Use the shortest possible wire length.
- Use wire trays for routing where possible.
- Avoid running wires near high-energy wiring.
- Avoid running input wiring in close proximity to output wiring.
- Avoid creating sharp bends in the wires.

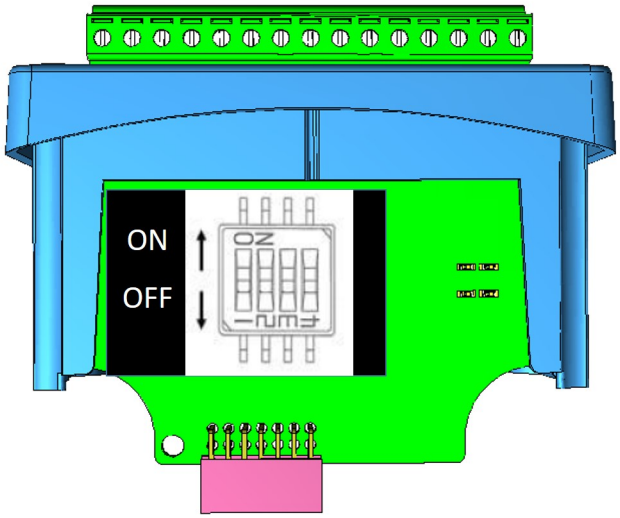
3.3.3 Supported I/O Modules

Prepare a Torx T8 screwdriver for I/O modules installation

3.3.3.1 WISE-S214 (4AI/4DI)

Analog Input

- **Channels:** 4
- **Resolution:** 16bits Bipolar; 15bits Unipolar
- **Sampling Rate:** 10Hz (Total) with50/60Hz Rejection
- **Accuracy:** ±0.1% for Voltage Input; ±0.2% for Current Input
- **Input Range:** 0~150mV, 0~500mV, 0~1V, 0~5V, 0~10V, ±150mV, ±500mV, ±1V, ±5V, ±10V, 0~20mA, ±20mA, 4-20mA
- **Input Impedance:** >1MΩ (Voltage)
  240 Ω (current)
- **Support Data:** Max/min, Scaling and Averaging
- **Support burn out detection**



DI Switch	Status	Condition
SW1 (Vo0)	ON	Current Input
	OFF	Voltage Input
SW2 (Vo1)	ON	Current Input
	OFF	Voltage Input
SW3 (Vo2)	ON	Current Input
	OFF	Voltage Input
SW4 (Vo3)	ON	Current Input
	OFF	Voltage Input

Digital Input

- **Channels:** 4 Dry Contact (Wet Contact by request for customization)
- **Logic Level:**
  - 0: Open
  - 1: Close to DI COM
- **Channel Mode:** DI (Logic status, Counter, Low to High Latch, High to Low Latch, Frequency
- **Compatibility:** 3.3V/TTL
- **Supports 200Hz Counter Input (32-bit + 1-bit overflow)**
- **Supports keep/discard counter value on power-off**
- **Support inverted digital input status**
- **Support configuration by each channel**
- **Support digital filter (min 0.1ms)**
- **Support high-to-low and low-to-high latch**

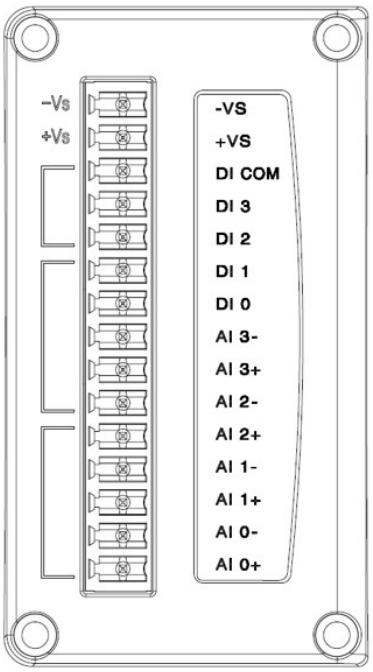


Figure 3.11 WISE-S214 Pin Assignment

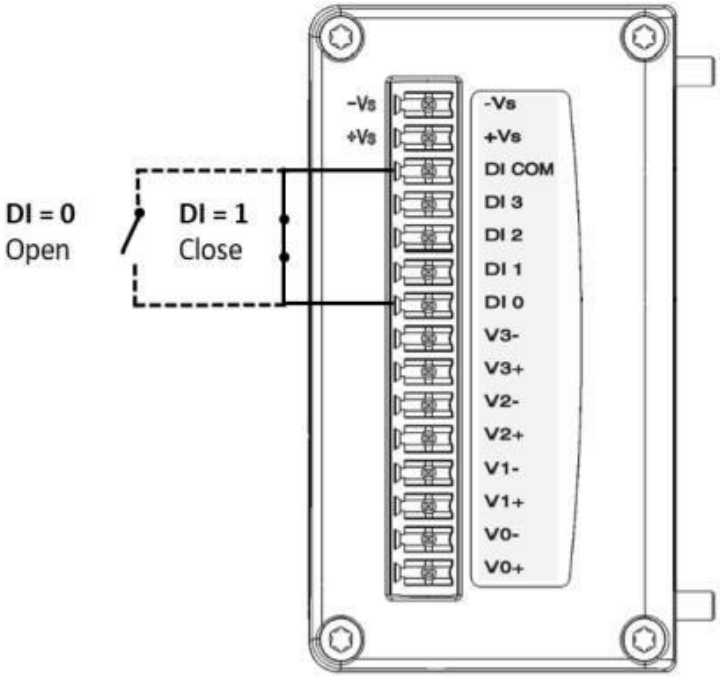


Figure 3.12 WISE-S214 Digital Input Wiring Diagram

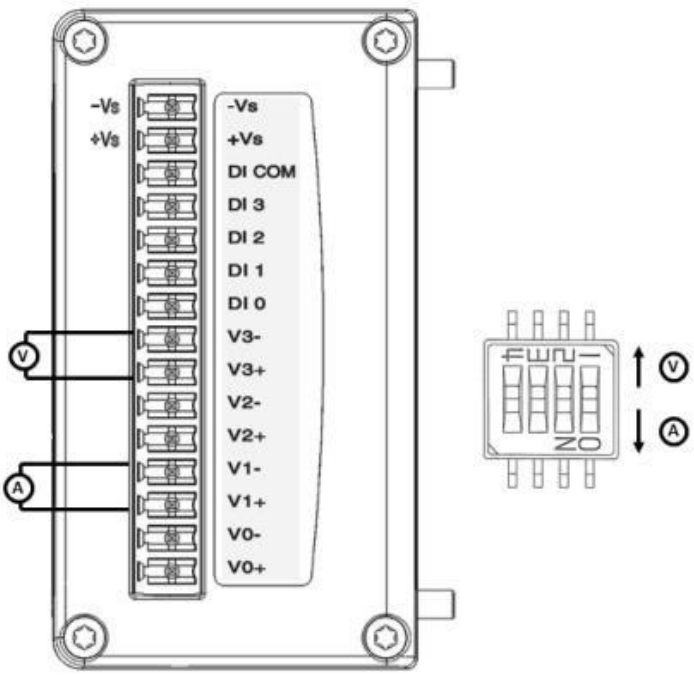


Figure 3.13 WISE-S214 Analog Input Wiring Diagram

3.3.3.2 WISE-S250 (6DI, 2DO& 1RS-485)

Digital Input

- **Channels:** 6 Dry Contact (Wet Contact by request for customization)
- **Logic Level:**
  - 0: Open
  - 1: Close to DI COM
- **Compatibility:** 3.3V/TTL
- **Supports 3kHz Frequency Input**
- **Supports 3kHz Counter Input (32-bit + 1-bit overflow)**
- **Supports keep/discard counter value on power-off**
- **Support inverted digital input status**
- **Support configuration by each channel**
- **Support digital filter (min 0.1ms)**
- **Support high-to-low and low-to-high latch**

Digital Output (Sink Type)

- **Channel:** 2
- **Output Current:** 100 mA
  At 0 -> 1: 100 us
  At 1 -> 0: 100 us
  (for Resistive Load)
- **Channel Mode:** DI (Logic status, Counter, Low to High Latch, High to Low Latch, Frequency
- **Supports Pules Output:** 5 kHz
- **Max. Load Voltage:** 30V
- **Support pulse high/low width and duty cycle adjustment**
- **Support high to low and low to high delay time setup**

Serial Port

- **Port Number:** 1
- **Type:** RS-485
- **Data Bits:** 8
- **Stop Bits:** 1, 2
- **Parity:** None, Odd, Even
- **Baud Rate (bps):** 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
- **Protocol:** Modbus/RTU (Total 64 addresses by 30 max. instructions)
- **Support Server response timeout and Delay between Polls setting**

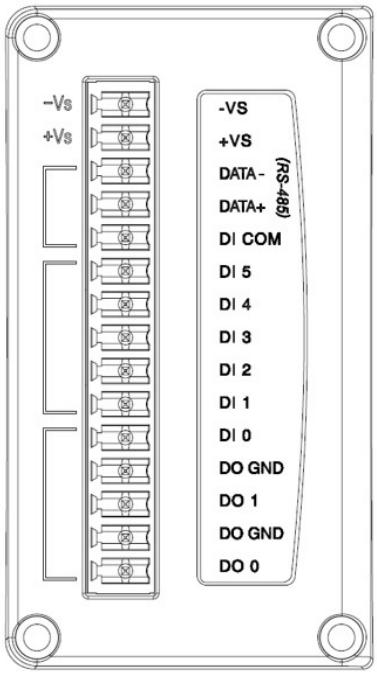


Figure 3.14 WISE-S250 Pin Assignment

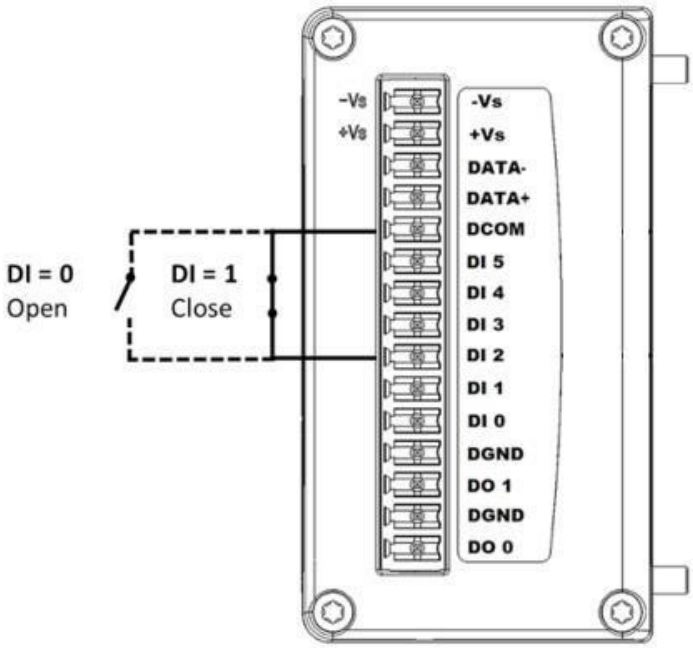


Figure 3.15 WISE-S250 Digital Input Wiring Diagram

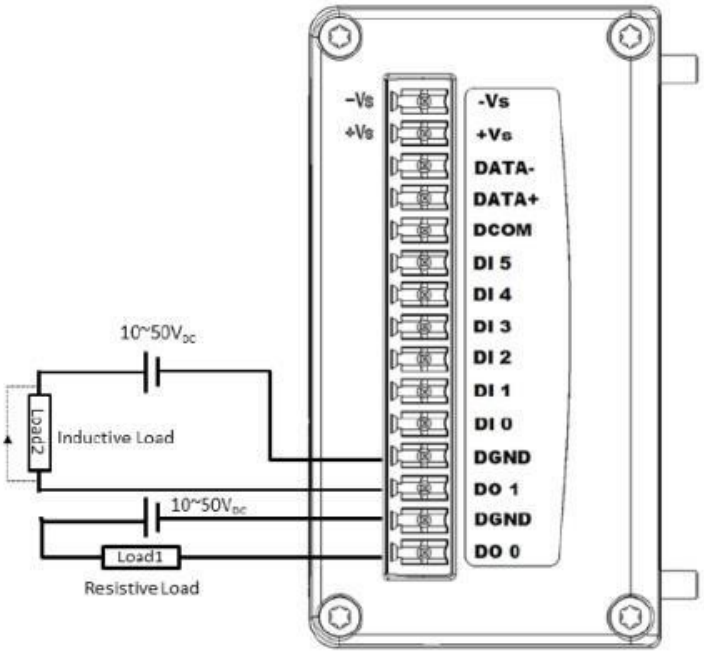


Figure 3.16 WISE-S250 Analog Input Wiring Diagram

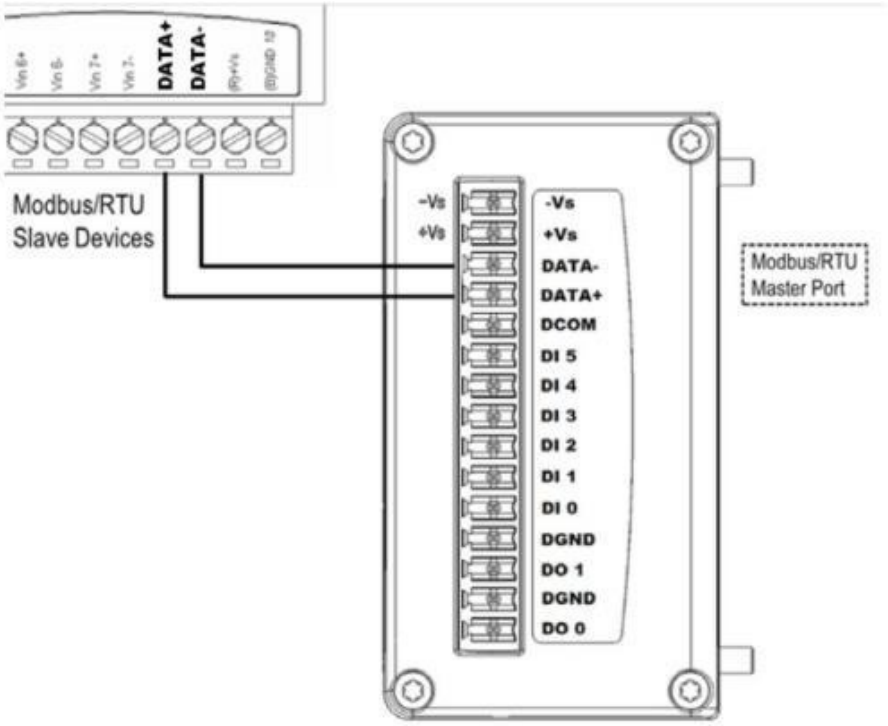


Figure 3.17 WISE-S250 RS-485 Wiring Diagram

3.3.3.3 WISE-S251 (6DI/1RS-485)

Digital Input

- **Channels:** 6 Dry Contact (Wet Contact by request for customization)
- **Logic Level:**
  - 0: Open
  - 1: Close to DI COM
- **Channel Mode:** DI (Logic status, Counter, Low to High Latch, High to Low Latch, Frequency
- **Compatibility:** 3.3V/TTL
- **Supports 200Hz Counter Input (32-bit + 1-bit overflow)**
- **Supports keep/discard counter value on power-off**
- **Support inverted digital input status**
- **Support configuration by each channel**
- **Support digital filter (min 0.1ms)**
- **Support high-to-low and low-to-high latch**

Serial Port

- **Port Number:** 1
- **Type:** RS-485
- **Data Bits:** 8
- **Stop Bits:** 1, 2
- **Parity:** None, Odd, Even
- **Baud Rate (bps):** 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
- **Protocol:** Modbus/RTU (Total 64 addresses by 30 max. instructions)
- **Support Server response timeout and Delay between Polls setting**

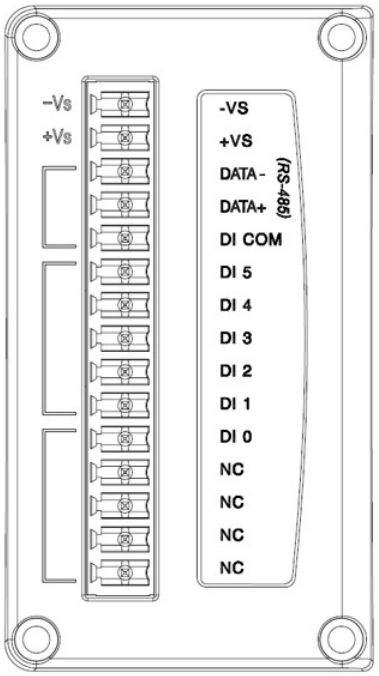


Figure 3.18 WISE-S251 Pin Assignment

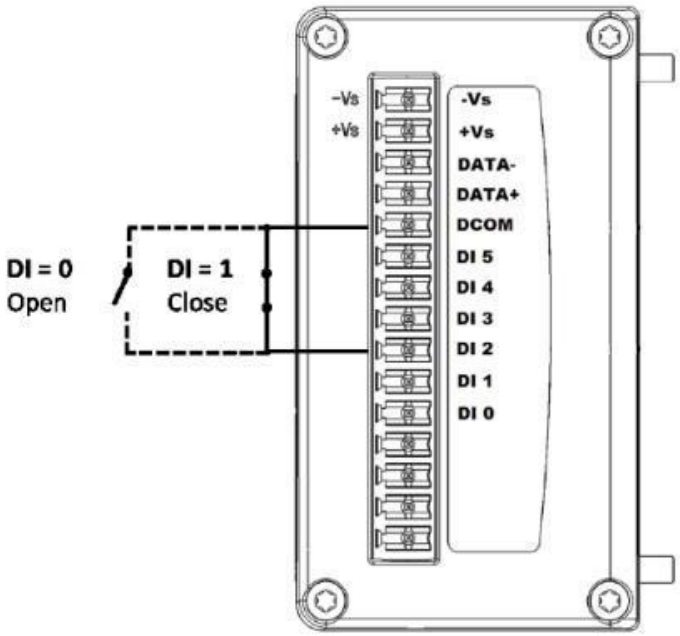


Figure 3.19 WISE-S251 Digital Input Wiring Diagram

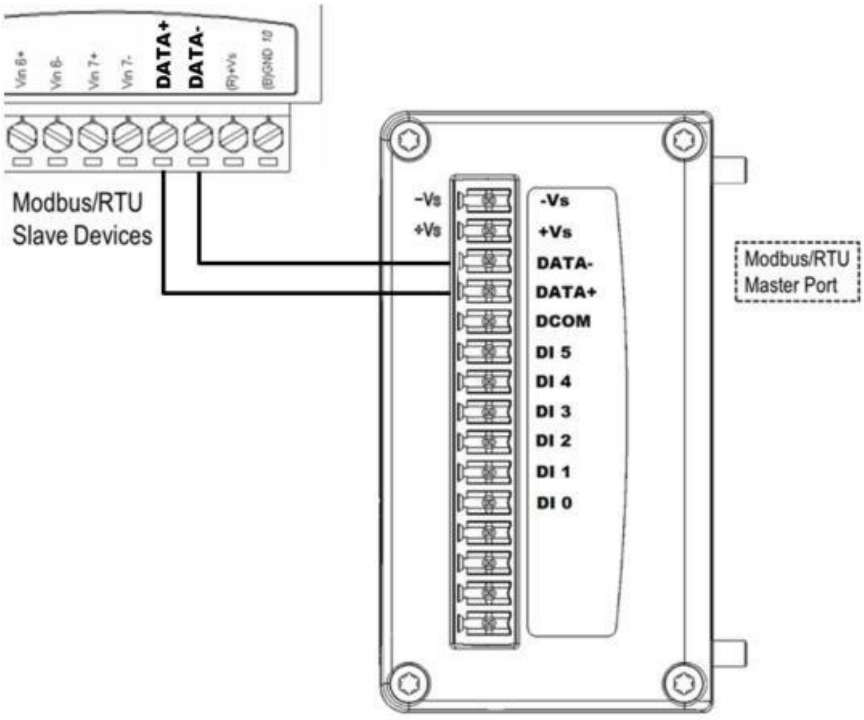


Figure 3.20 WISE-S251 RS-485 Input Wiring Diagram

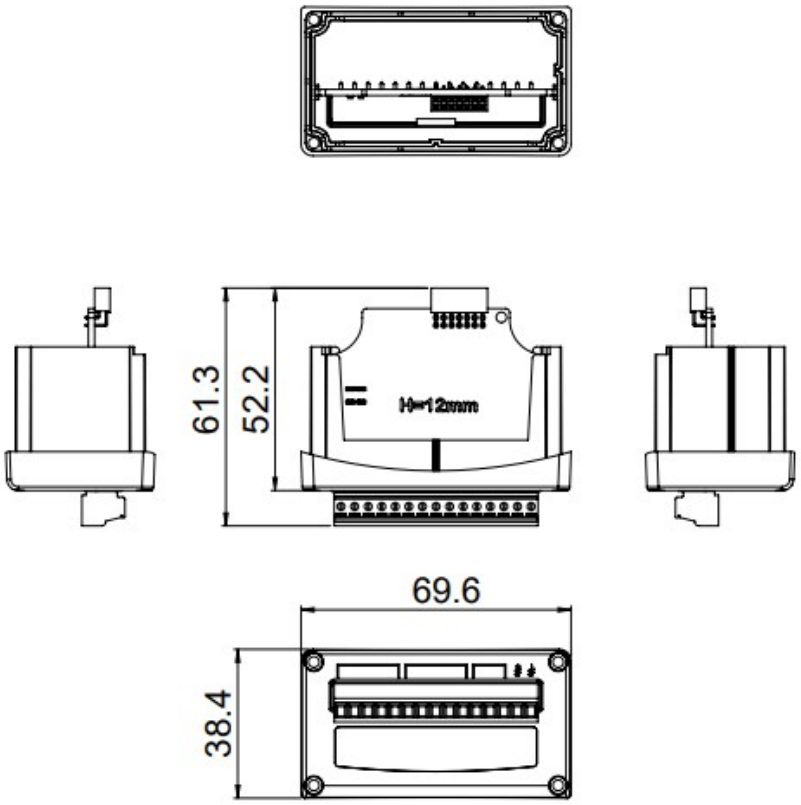
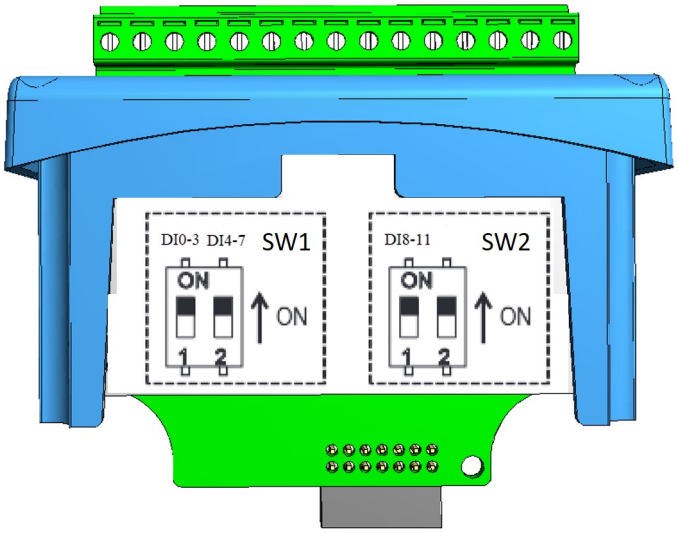


Figure 3.21 Dimension of WISE-S214/S250/S251

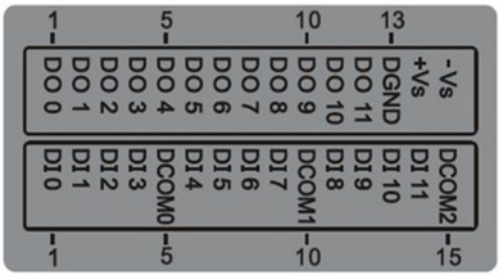
3.3.3.4 WISE-S252 (12DI/12DO)

Digital Input

- **Channels:** 12
- **Logic Level**
  - Dry Contact 0: Open
    1: Close to DCOM
  - Wet Contact 0: -5~5 V_{DC}
    1: -17~-30 V_{DC} or 17~30 V_{DC} (2 mA min.)
- **Input Voltage:** 50 V_{DC} max
- **Isolation:** 3,000 Vrms
- **Channel Mode:** DI (Logic status, Counter, Low to High Latch, High to Low Latch, Frequency
- **Supports 1kHz Counter Input (32-bit + 1-bit overflow)**
- **Supports keep/discard counter value on power-off**
- **Support inverted digital input status**
- **Support digital filter (min 0.1ms)**
- **Support high-to-low and low-to-high latch**
- **Switch Label:**
  - ON: Dry Contact
  - OFF: Wet Contact

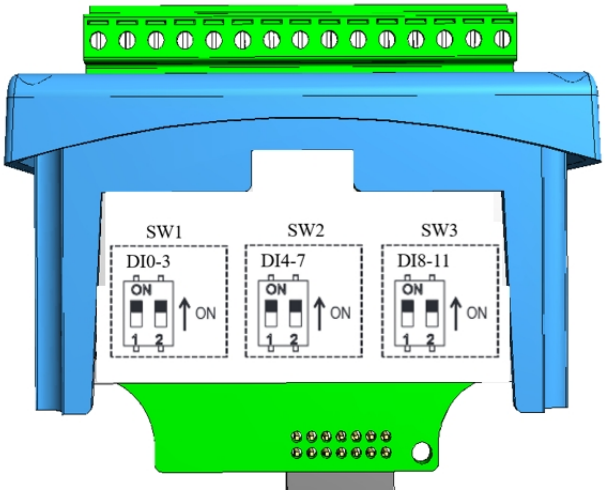


- **I/O Label:**

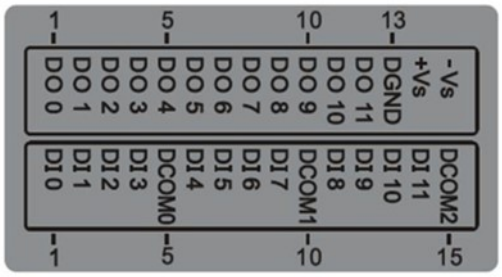


Digital Output (Sink Type)

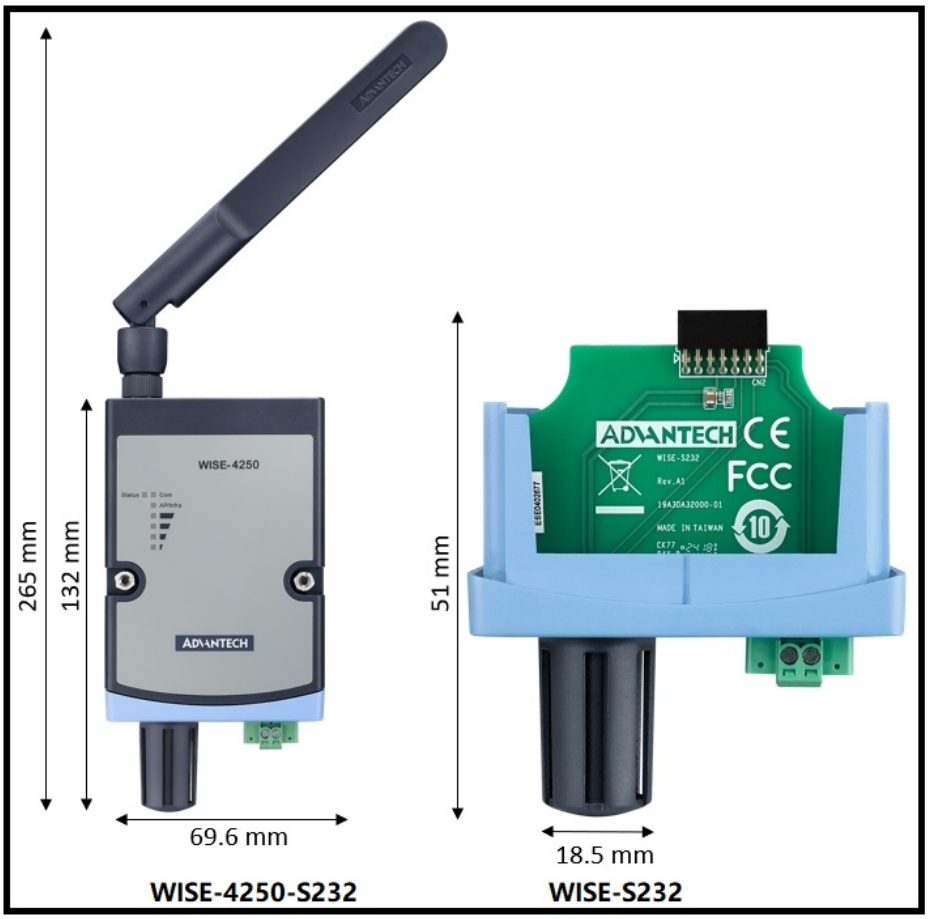
- **Channel:** 12 (Non-Isolation)
- **Output Current:** 100 mA
  - At 0 -> 1: 100 us
  - At 1 -> 0: 100 us
    (for Resistive Load)
- **Supports Pules Output 5 kHz**
- **Max. Load Voltage:** 30V
- **Support pulse high/low width and duty cycle adjustment**
- **Support high to low and low to high delay time setup**



DI Switch	Status	Condition
SW1-1	ON	DI 0-3 Dry Contact
SW1-2	OFF	DI 0-3 Wet Contact
SW2-1	ON	DI 4-7 Dry Contact
SW2-2	OFF	DI 4-7 Wet Contact
SW3-1	ON	DI 8-11 Dry Contact
SW3-2	OFF	DI 8-11 Wet Contact



3.3.3.5 WISE-S232 (Temperature & Humidity Sensor)



Temperature

- **Operating Range:** -25 °C ~ 70 °C (77 °F ~ 158 °F)
- **Update Rate:** Min. 1 second, Max. 24 hours (with WISE-4250)
- **Resolution:** 0.01 (°C)
- **Accuracy:** ±1°C (at 25 °C)
- **Response time:** 2 seconds (at 25 °C and 1m/s airflow)
- **Long Term Drift:** <0.04°C/year

Humidity

- **Operating Range:** 0 ~ 100% RH (Recommended 20~80% RH)
- **Update Rate:** Min. 1 second, Max. 24 hours (with WISE-4250)
- **Resolution:** 0.01% RH
- **Accuracy:** ±4% RH (at 25°C) @ 0%~90% RH ±5% RH (at 25°C) @ 90%~100% RH
- **Response time:** 6 seconds (at 25°C and 1m/s airflow)
- **Long Term Drift:** <0.5%RH/year

* Default value of measurement interval is 15 seconds, user can set in the configuration page.

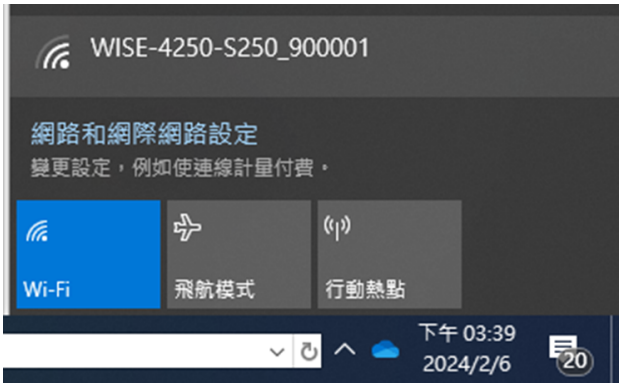
* Users can independently procure and replace accessories (cap and ePTFE membrane) to enhance protection in harsher environments.

Chapter 4

System Configuration

4.1 Connection

1. Plug a DC power source into the +Vs, -Vs pin of your module to turn on the power.
2. When users set the switch to “ON” as Section 1.3 shows, the module will be set to normal mode and the default setting for this operation mode is AP mode. If you want to set the module to Initial Mode (i.e. factory default mode), you can change position 1 of SW1 to “OFF”. The module will only work in AP Mode when the module is in Initial Mode. Now the module can be searched by mobile devices or wireless adapter of computer with SSID: WISE-4250_I/O module name_MAC Address. Click the SSID to connect the module in AP Mode, WISE module will auto assign the IP address for mobile devices or computer.



4.2 Configure WISE Using Web Interface

4.2.1 System Requirements

The web utility of WISE-4250 module is developed with public HTML 5, but for detailed indication and data transmission modes, the type of web utility will depend on web page of the operating system. For mobile devices, the minimum system requirements of web browsers are as below:

- Safari 6 in Apple iOS
- Web Browser in Google Android 4.0 (Ice Cream Sandwich)
- Chrome in Google Android 4.0 (Ice Cream Sandwich)

Table 4.1:

Mobile Browser	Chrome	Android	Safari
Configuration	Y	Y	Y
File Upload	N	N	N
Data Log Chart	Y	Y	Y
Data Log Export	N	N	N

For PC platforms, the minimum requirements of web browsers are as below:

- Internet Explorer (version 11)
- Google Chrome (version 30)
- Mozilla Firefox (version 25)

Table 4.2:

Mobile Browser	Chrome	Firefox	Safari	IE11	IE10
Configuration	Y	Y	Y	Y	Y
File Upload	Y	Y	N	Y	N
Data Log Chart	Y	Y	Y	Y	Y
Data Log Export	Y	Y	N	N	N

4.2.2 Factory Default Settings

WISE-4250 Series

- Operation Mode: Normal Mode
- Wireless Mode: AP Mode
- IP Mode: Static IP Address
- Default IP: 192.168.1.1
- Subnet Mask: 255.255.255.0
- Default Gateway/DNS: 192.168.1.1
- DHCP Server: Enabled
- Default Connection Timeout: 720 seconds
- HTTP Port: 80

4.2.3 Module Authorization

Table 4.3:

Account	Default Password	Access Ability
root	00000000	All privileges
admin	00000000	All privileges except access control configuration
user	00000000	View module status only. Not allow to change configurations

Table 4.4:

Functions	Account		
	root	admin	user
Device information	View	View	View
Device setting	Edit	Edit	Deny
System Restart	Edit	Edit	Edit
Module Locate	Edit	Edit	Edit
Change passwords	Edit	Deny	Deny
Reset Password	Edit	Deny	Deny
Reset to default	Edit	Deny	Deny
Access control configurations	Edit	Edit	Deny
Group configurations	Edit	Edit	Deny

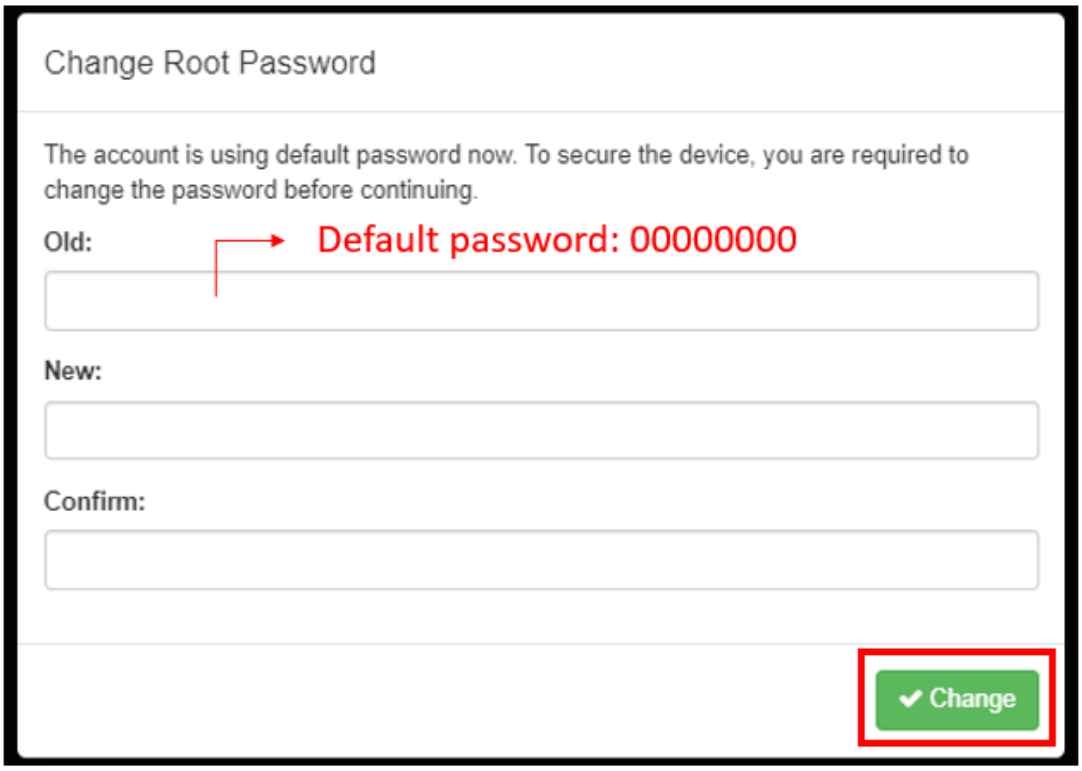
Download/upload processes	Edit	Edit	Deny
Network configurations	Edit	Edit	View
I/O configurations	Edit	Edit	View
I/O statuses monitor	View	View	View
Reset AI calibration to default	Edit	Deny	Deny
MODBUS addresses	Edit	Edit	View
Data log configuration and query	Edit	Edit	View
Clear data log	Edit	Edit	Deny

4.2.4 Logging Page

4.2.4.1 Change Root Password

To enhance system security and comply with the EU's Product Security and Telecommunications Infrastructure (PSTI) security regulations, please change the password value before logging to the configuration page.

*Please remember the new password you set. New password cannot be the same as default password value (00000000).



4.2.5 Operation Mode

The operation mode can be configured by switch SW1 on the back of module. Please refer to previous chapter for the detail of configuring SW1.

Table 4.5: Operation Mode

Mode	WISE-4250 Series
Initial Mode	AP mode. Fixed IP address: 192.168.1.1 Fixed Wi-Fi Mode: AP Mode
Normal Mode	AP mode or Infrastructure mode. Default IP address: 192.168.1.1 Default Wi-Fi Mode: AP Mode

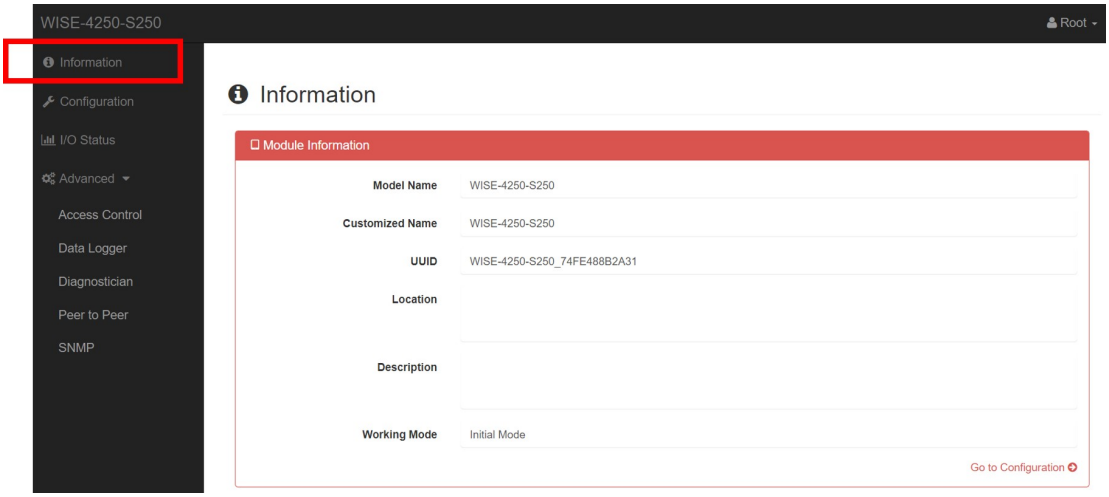
- **AP Mode (Only 1 Connection)**
  When WISE-4250 work in Limited AP mode, user can find the SSID for WISE module, and connecting to it as a wireless switch. It makes the configuration and diagnostic of WISE module much easier.
- **Infrastructure Mode (4 Connections)**
  In general, WISE modules stay connected to access point (AP) to be online. Users who want to connect their mobile devices to WISE modules will need to connect to the same AP as WISE modules connected.

4.2.6 Using Web Browser to Configure the Module

- Configure URL: http://IP_address/config
- Default URL: http://192.168.1.1/config
- Configuration Steps

4.2.6.1 Module Information

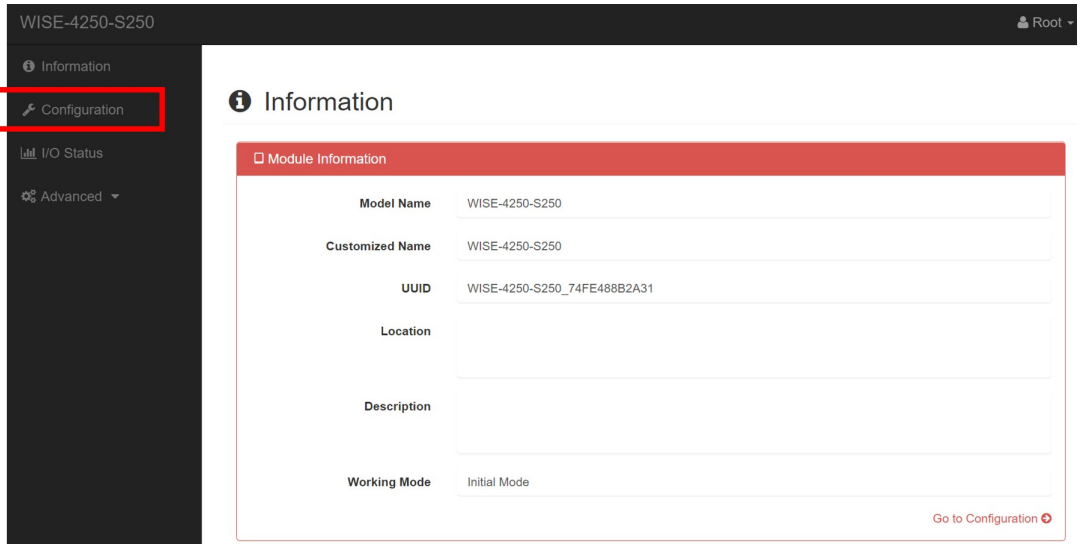
In the information page, you can see the dashboard: Module information, Wireless status, Network information and firmware version.



4.2.6.2 Module

You can see the naming of the module and related information. Click “Configuration” to view or change the configurations:

- **Model Name:** Model name of the WISE module
- **Customized Name / UUID:** Refer to Model name and UUID of the module. The default UUID is the combination of model name and the MAC address. This can be modified.
- **Location Information:** The location of the module can be stated here.
- **Description:** Any comments about this module can be stated here.
- **Working Mode:** Refer to 4.2.4 for Operation Mode

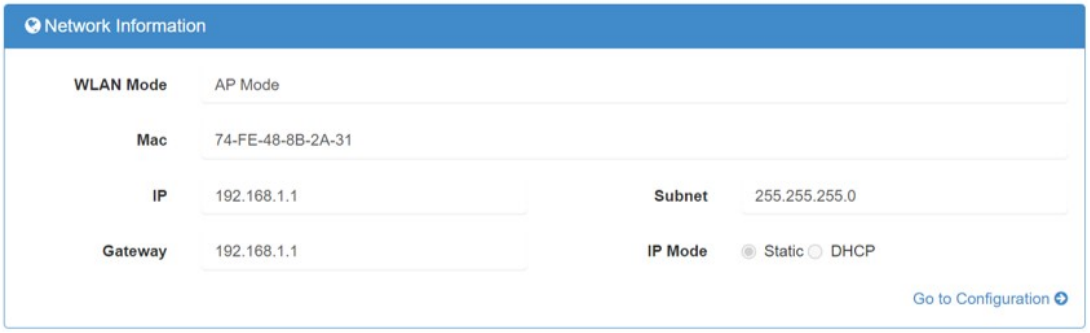


4.2.6.3 Wireless Status

For WISE-4250 series, users can refer to the WLAN RSSI indicators for the signal quality in Wireless Status section. It also shows the MAC ID of the client device. If the module is in AP Mode, the WLAN RSSI Level and Refresh button will not be shown.

4.2.6.4 Network Information

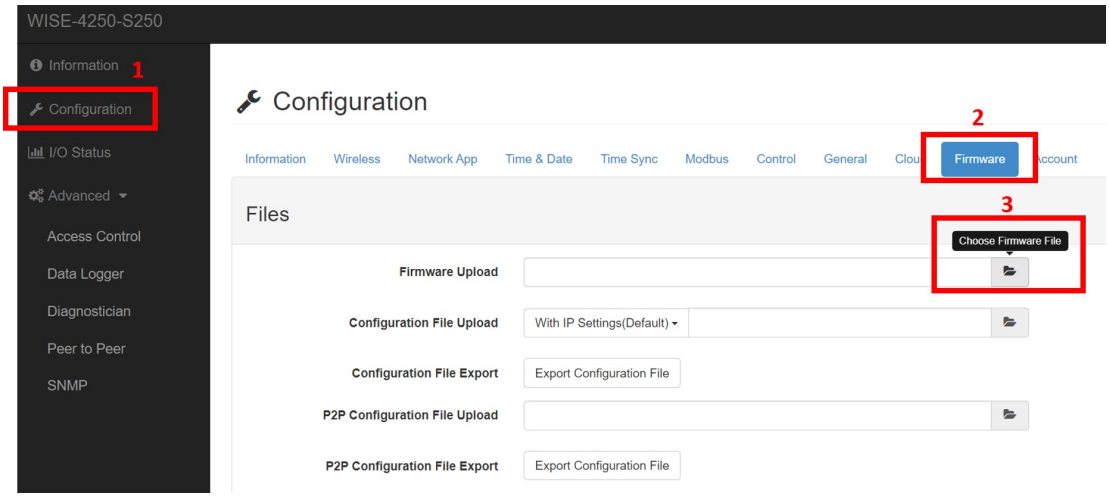
For WISE-4250 series, WLAN Mode (AP Mode / Infrastructure) will be shown in Network Information section. Here is an overview of the entire network configuration. To configure the network configuration, click “Configuration”.



4.2.6.5 Module Information

Here you can check the model name and the module description. The firmware version will also been shown here. At the end of the configuration web page, you can see the version. These versions have to be updated simultaneously.

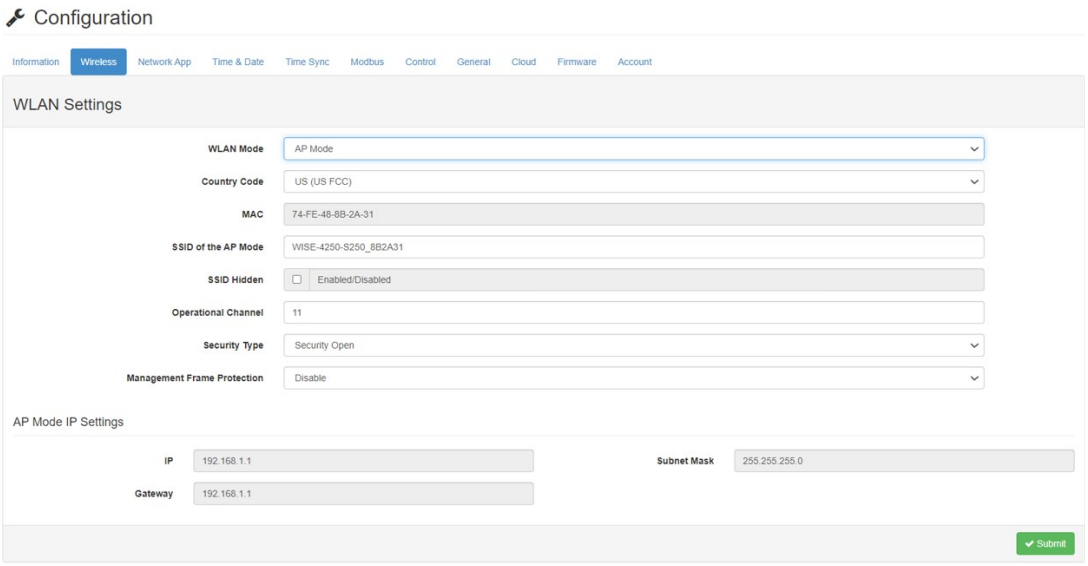
If you want to update the firmware version, go the "Configuration --> Firmware" section.



4.2.6.6 Wireless Setting Configuration

- **AP Mode**

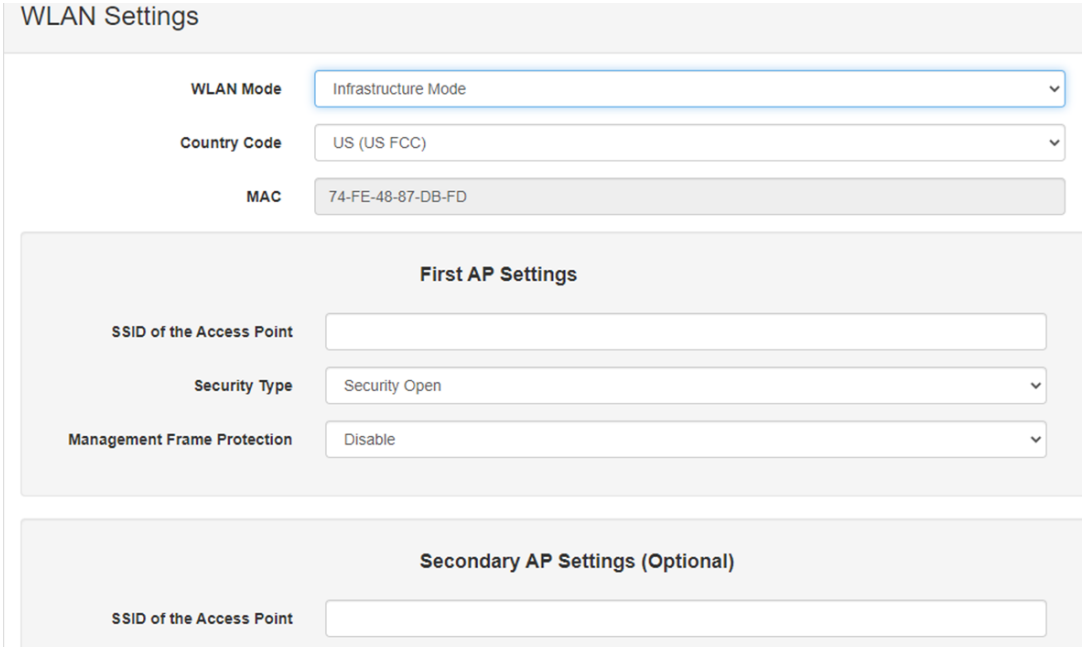
  When setting the module to AP mode, users can configure SSID, and choose country code, operation channel and security type.The “AP Mode IP Settings” are fixed and do not allow users to make changes.



- **Infrastructure Mode**

  When using the module in “Infrastructure Mode”, users need to enter the SSID of the Access Point (AP) that the WISE module is going to connect, and configure the security type here. WISE-4250 module provides country code selection and “Second AP Setting” section, an optional setting for the WISE module to connect to another AP automatically. If you do not have a second AP, just leave the SSID blank.

  After configuring the AP that the WISE module wants to access, the IP address also needs to be defined in the Infrastructure-Network. Users can choose to lock BSSID or not.



On the other hand, if you don't have a need for IP setting, you can choose "DHCP" mode in Infrastructure Mode IP Setting.

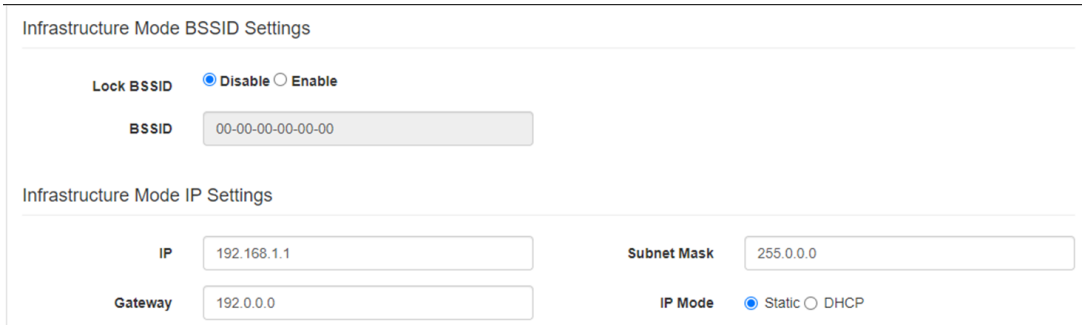
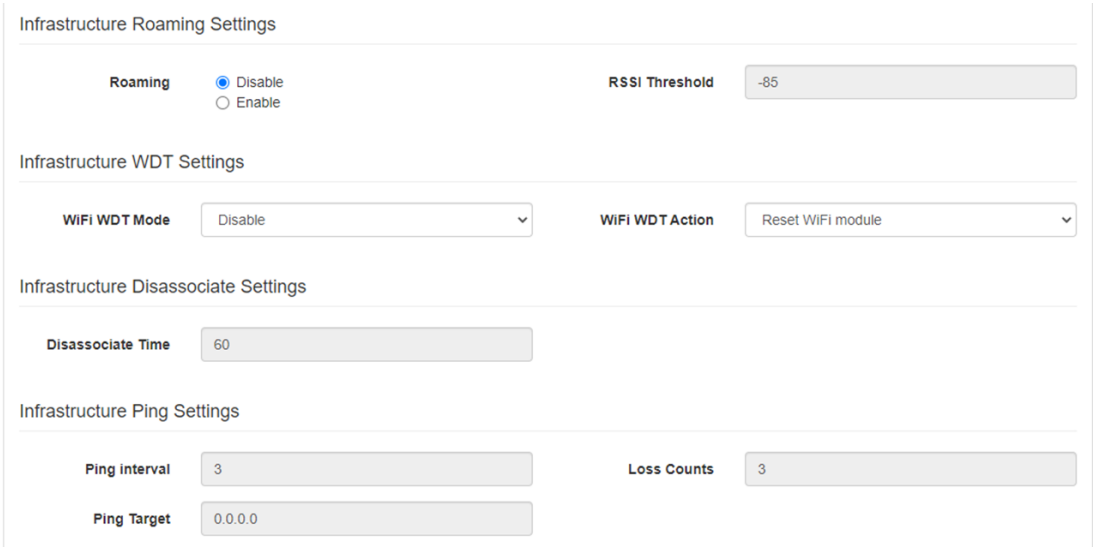
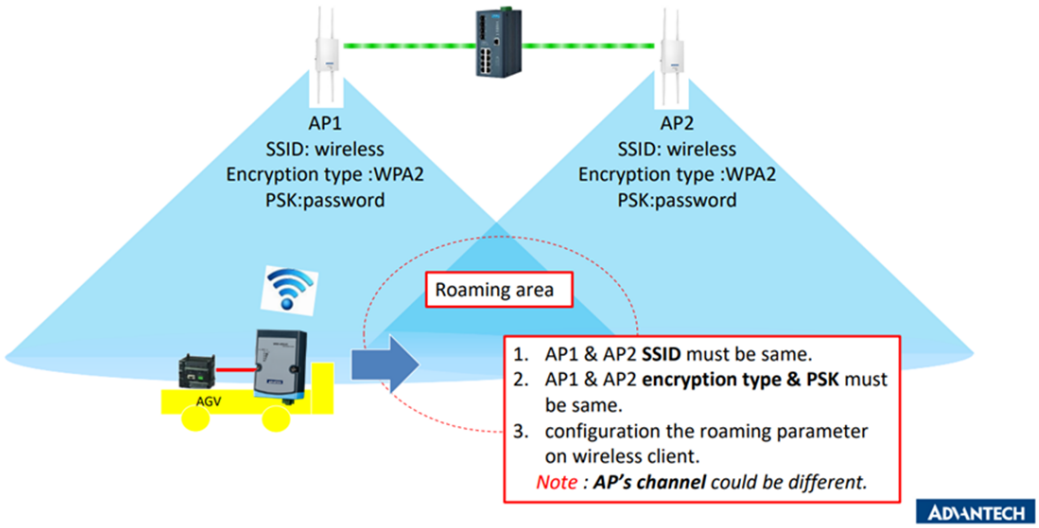


Table 4.6: Country Code Table		
Region	**2.4G**	**5G**
US	1 – 11	36 – 64 100 – 140 149 – 165
CA	1 – 11	36 – 64 100 – 116 132 – 140 149 – 165
SG	1 – 11	36 – 64 100 – 140 149 – 165
EU	1 – 13	36 – 64 100 – 140
AU	1 – 13	36 – 64 100 – 140
KR	1 – 13	36 – 64 100 – 140
FR	1 – 13	36 – 64 100 – 140 149 – 165
JP	1 – 14	36 – 64 100 – 140
CN	1 – 13	149 – 165

WISE-4250 provide new network configuration settings such as roaming setting, WDT setting, Disassociate Settings and Ping Settings.

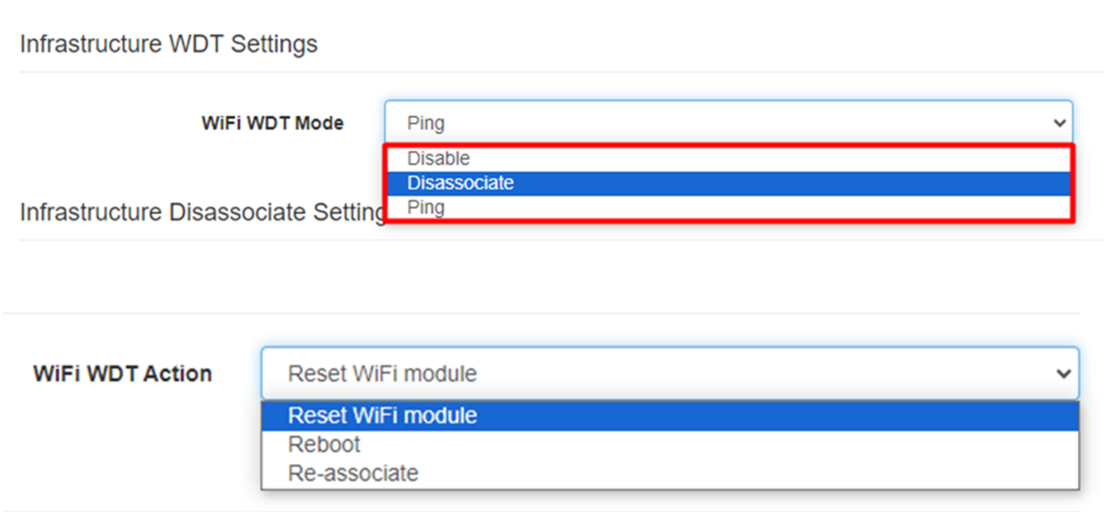


For Roaming Settings, WISE-4250 will do roaming when the RSSI of connected AP dropping to a threshold (can be set). Then WISE-4250 will start to find a new AP which has stronger signal. This is different from BSSID function of WISE-4000.WISE-4250 default support 802.11r roaming mechanism.

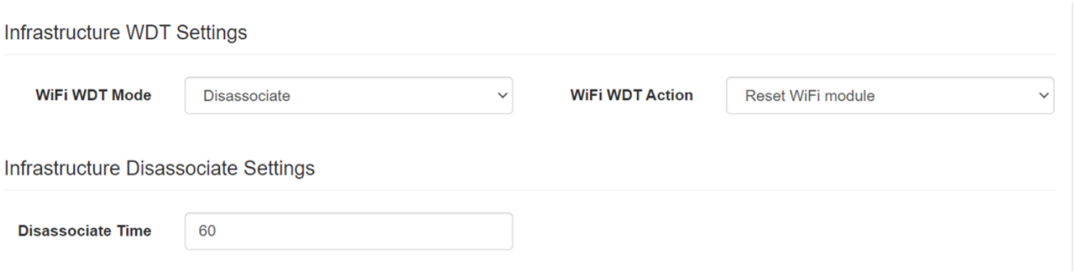


On the other hand, WISE-4250 offers two WiFi WDT Mode for user to configure how to trigger the WDT action and choose the specific action when the WDT had been trigger.

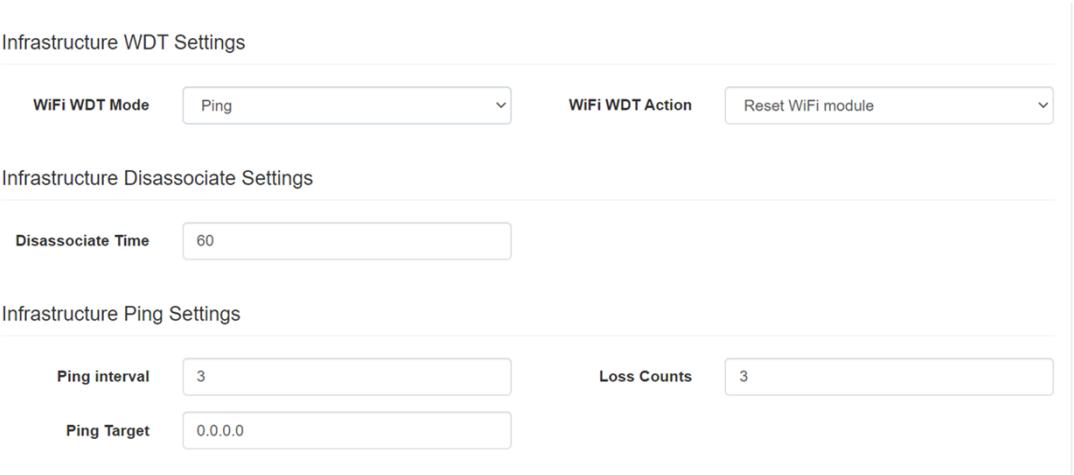
- In Disassociate mode,



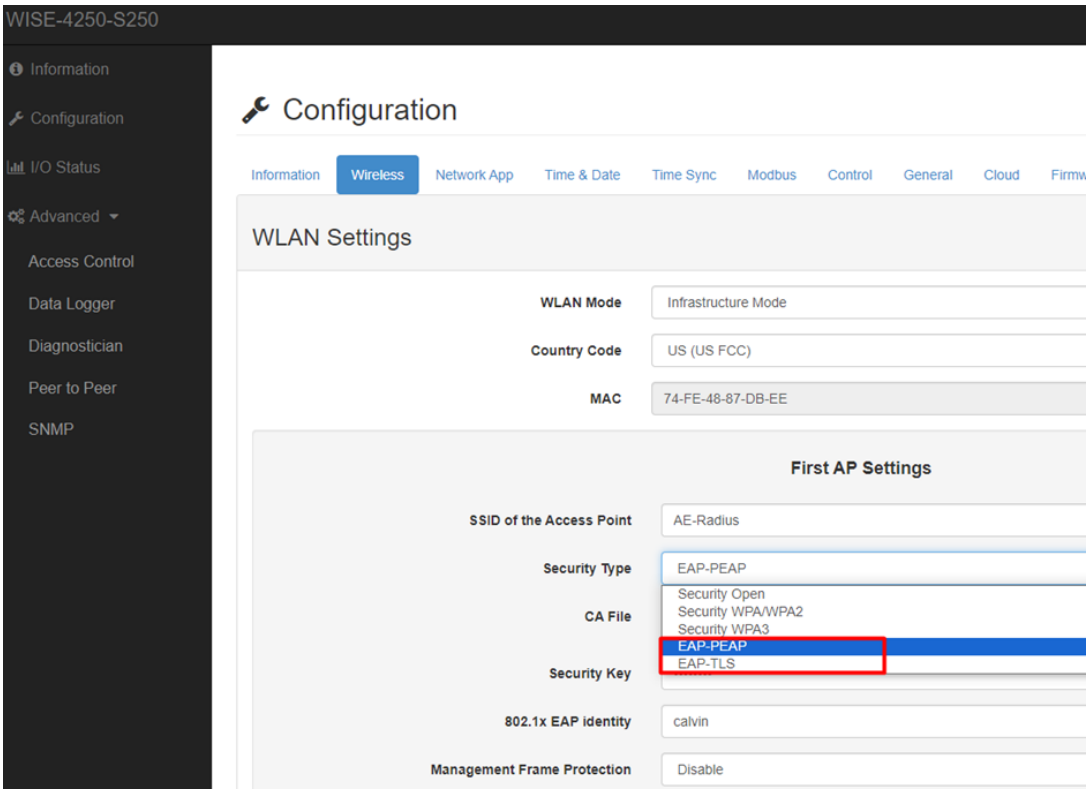
Disassociate mode refers to the disconnection or disassociation between the device and the network. WISE-4250 will trigger WDT action when it disconnected with AP over the Disassociate Time. This may occur due to network issues, communication failures, or problems with the device itself, resulting in the device losing its connection to the network. In Disassociate mode, the device needs to attempt to reestablish the connection to re-associate with the network and restore normal communication.



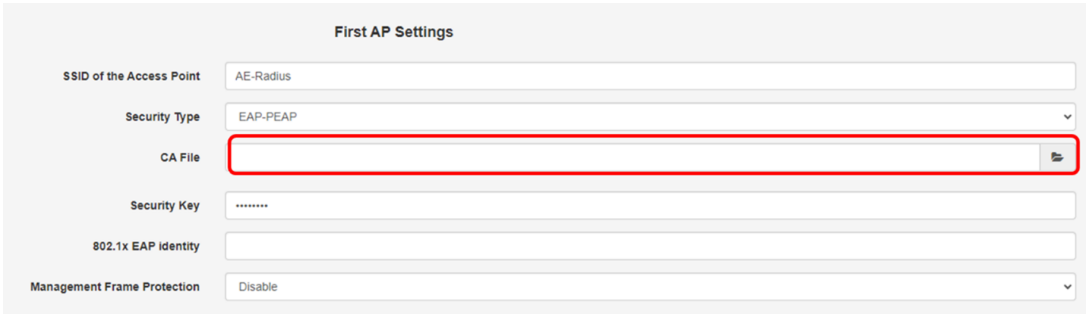
In Ping mode, WISE-4250 periodically sending ping messages to a server to confirm its online status. Upon receiving the ping message, the server can reply with a confirmation message, allowing the WISE-4250 to know it's still connected to the server. This helps the WISE-4250 maintain its connection status and ensures the reliability of data transmission.



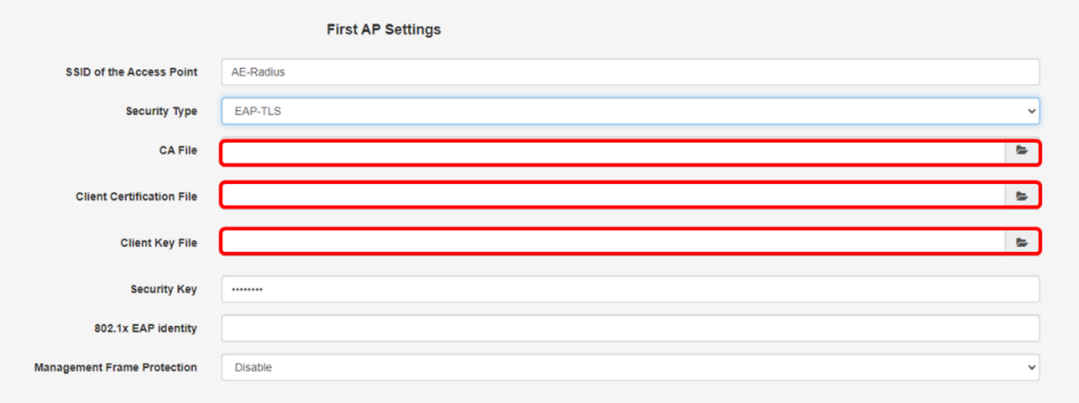
WISE-4250 support WPA, WPA2 and WPA3, and Authentication Protocol of EAP-PEAP and EAP-TLS.

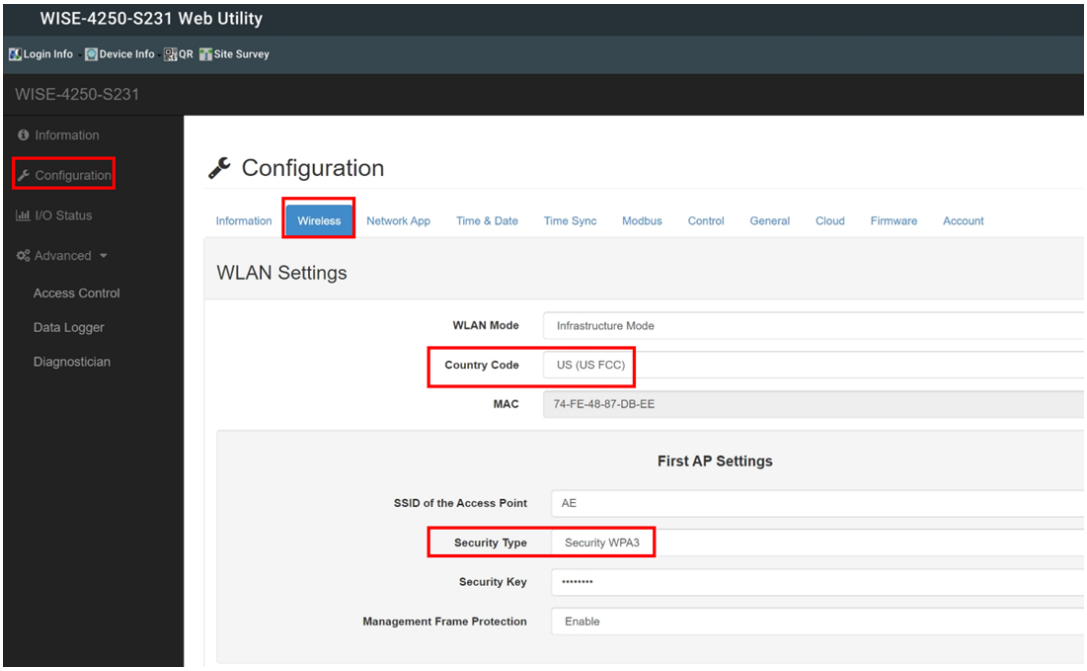


Enterprise with EAP-PEAP authentication protocol needs to upload the CA File to WISE-4250. This is the difference between WISE-4000 and WISE4250.



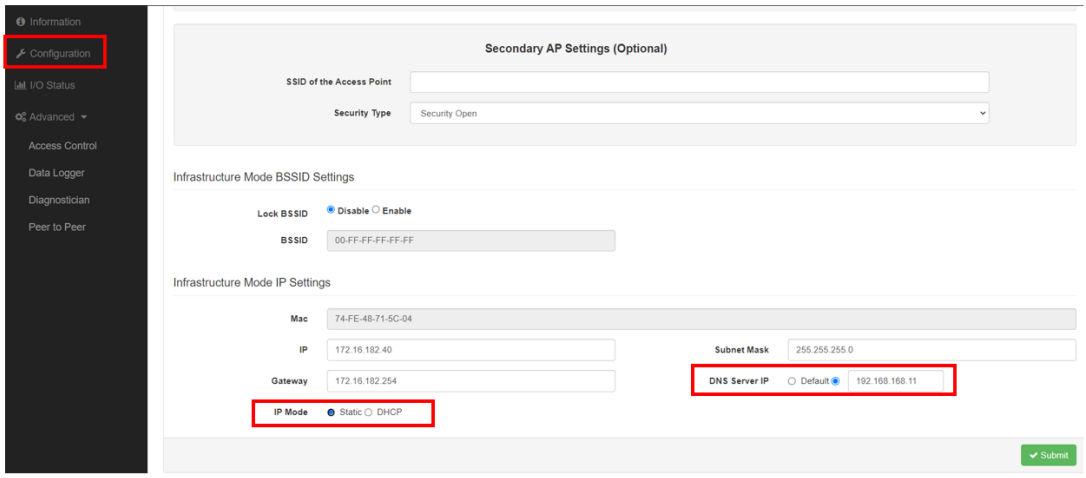
Country code should meet the setting of AP.TLS authentication protocol support up to TLS1.3, it needs to upload the CA File, Client Certification File and Client Key File to WISE-4250.





- DNS Server IP

(Configuration → Wireless → Infrastructure Mode IP Settings)
When the IP Mode is set to Static, the default value for the DNS Server IP is 1.1.1.1. When the IP Mode is set to DHCP, the IP, Subnet Mask, Gateway and DNS Server IP value can be modified by the user.

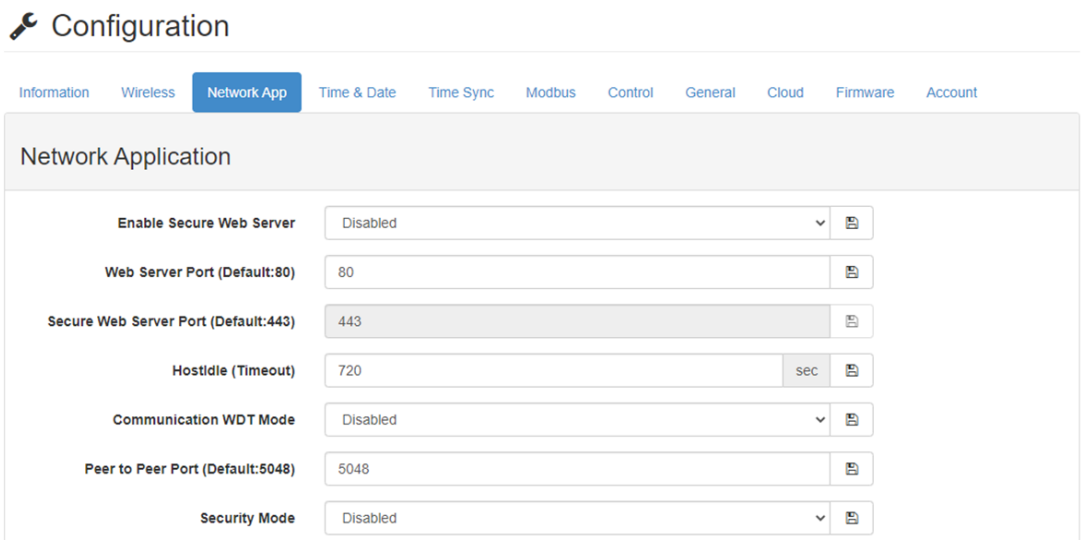


4.2.6.7 System Configuration

You can configure the Web Server Port, Host Idle (timeout), and decide whether to enable Communication WDT or not in the section. The “Web Server Port” field determines which Ethernet port will be used for the web service, such as the web configuration and RESTful Web API. The default port is 80, but if you change the port number to 8080, you will have to access the module through http://192.168.1.1:8080/config in AP mode.

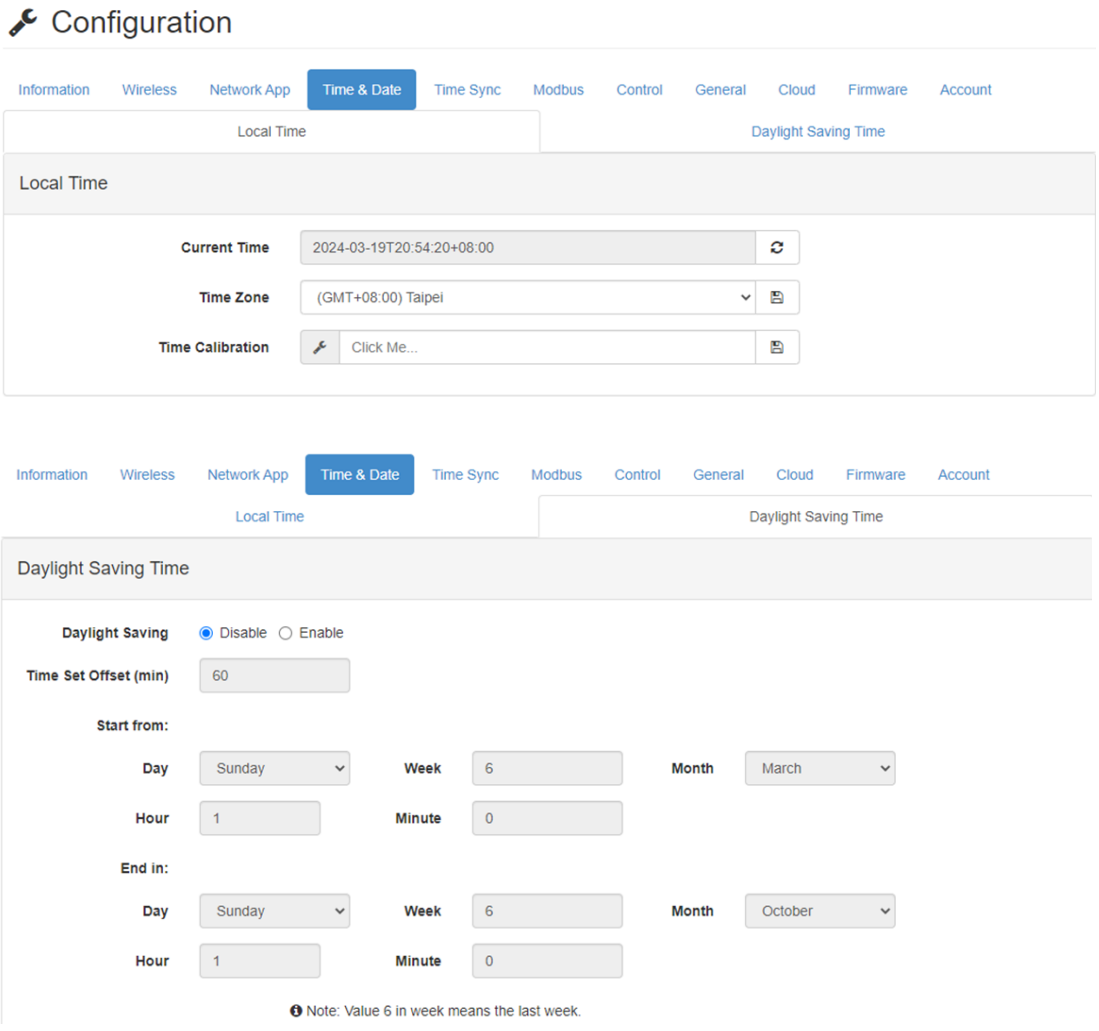
The **Host Idle (Timeout)** field dictates the availability of the TCP connection between the host controller and WISE. Each MCU-based WISE module supports four TCP connections from visitors simultaneously. In this case, if one of the hosts stops communicating with the WISE module for a time period longer than the configured host idle time (for example, the default 720 seconds), the WISE module will terminate the TCP connection with the host.

The Communication WDT has a Host Idle time, and will be triggered when all TCP connections are closed. This includes all hosts which visit WISE and the communication between remote servers like Dropbox or private servers. Once the **Communication WDT mode** is enabled, it will trigger system events, such as FSV of output channel and system log.



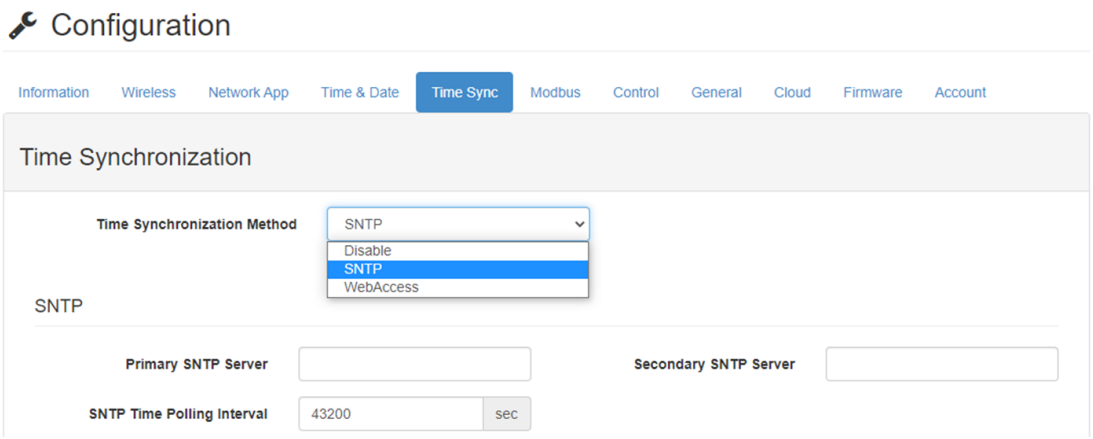
4.2.6.8 Time & Date

You can see the current time, select the time zone you want to use, and calibrate the time by clicking “Click Me” to inquire time from host devices. Users can also adjust daylight saving time.



4.2.6.9 Time Sync

You can enable SNTP, so that the module can act as an SNTP client to perform time synchronization from an assigned SNTP server. Or time sync with WebAccess.



4.2.6.10 Modbus Address

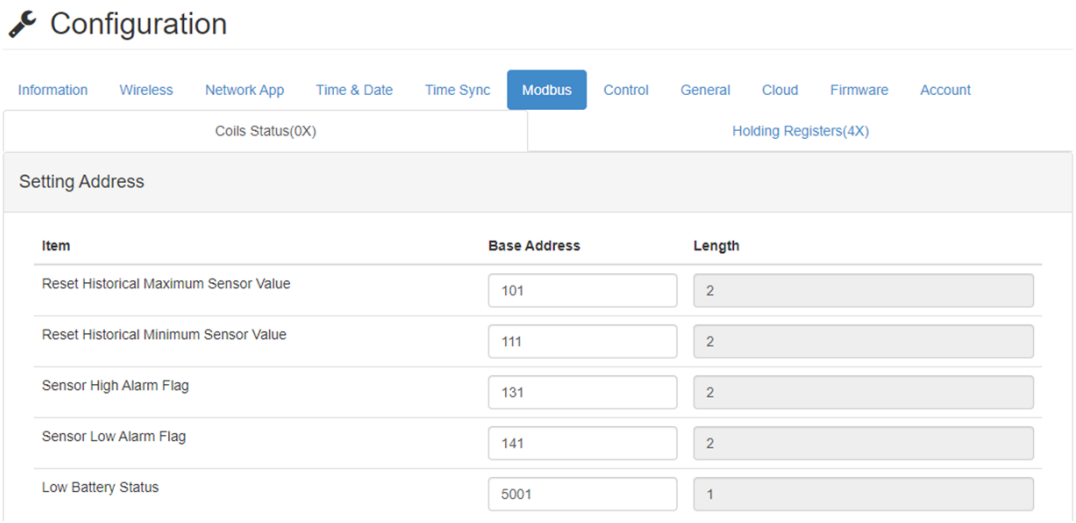
In order to provide flexibility and scalability for module deployment, we release the Modbus address setting table, so that you can configure based on your needs. The Modbus address sections are divided into two parts: 0X and 4X. These address sections correspond to different types of data registers, such as coils, discrete inputs, input registers, and holding registers. Here's what they mean:

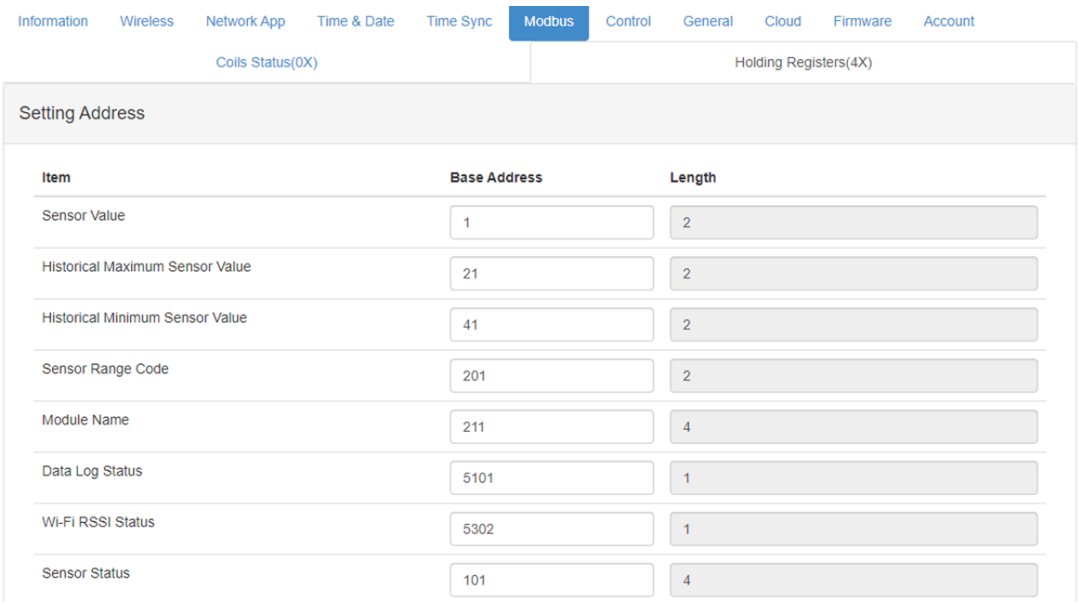
0X:

- This section represents coils and discrete inputs.
- Coils are read-write bits used for controlling outputs.
- Discrete inputs are read-only bits used to indicate the status of inputs, such as switch states.
- For example, 0X0001 might represent the first coil, while 0X1001 might represent the first discrete input.

4X:

- This section represents input registers and holding registers.
- Input registers are read-only and used to store input values, such as sensor data.
- Holding registers are read-write and used to store data or control values, such as device settings.
- For example, 4X0001 might represent the first input register, while 4X1001 might represent the first holding register.



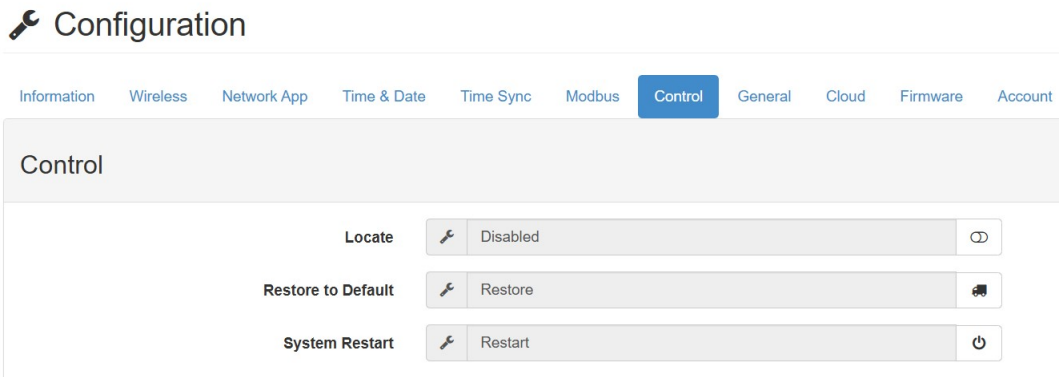


4.2.6.11 Control

Locate: Helps user search for lit modules. (The status LED will be on for 30 seconds when enabled.)

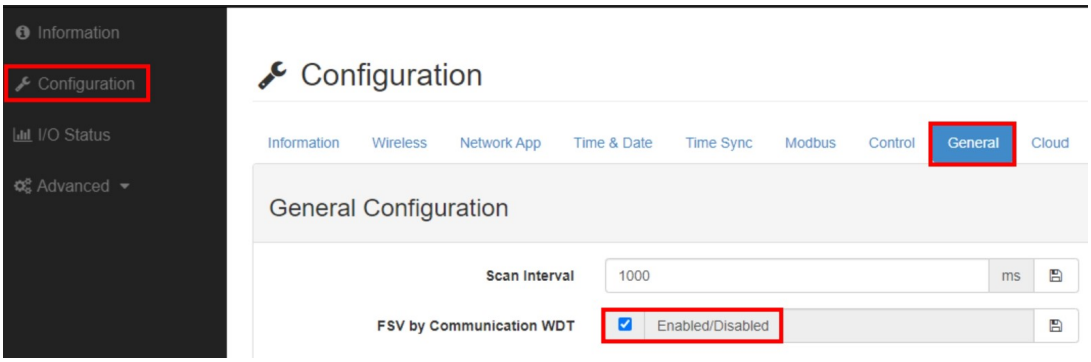
Restore to default: The system configuration will be cleared and restored to factory default settings when enabled.

System Restart: This module's system will reboot when enabled.



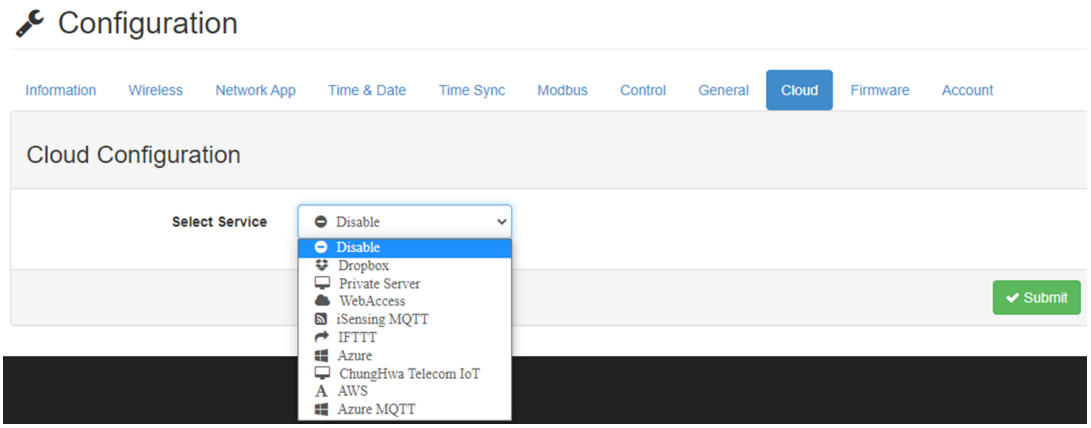
4.2.6.12 General Configuration

"**Scan Interval**" determines the I/O polling interval in the "I/O Status" section. The changed values will not be saved into the module. The change is only valid until the browser is closed. The "**FSV by Communication WDT**" determines whether to enable the FSV function triggered by communication with WDT. You can enable the FSV function for all the output channels of the module here.



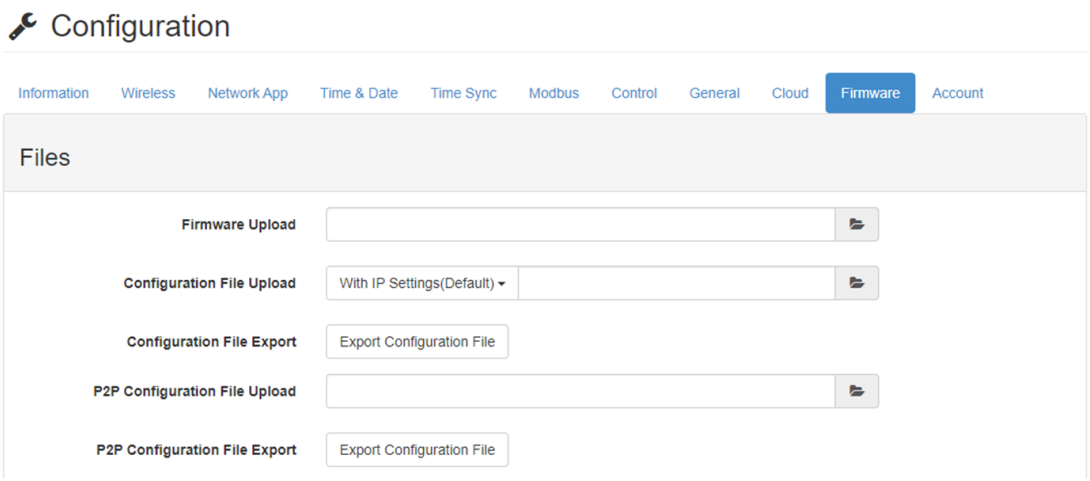
4.2.6.13 Cloud

User can choose supported cloud to connect with.



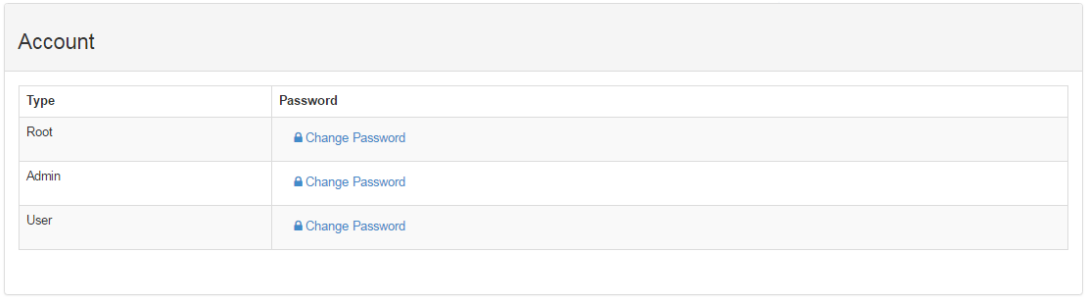
4.2.6.14 Firmware

Users can manage Firmware upload, Configuration File upload/export, P2P Configuration File upload/export here. If you want update the Firmware, click the icon to select the configuration file you want to upload from the local folder in “Firmware Upload”. However, You can find the latest official release firmware file at the Advantech support site (http://support.advantech.com/support/).



4.2.6.15 Account Management

Change the password for each account here.



Check Version

The firmware version is shown in the "System Information" page. At the end of the configuration web page, check the version. For normal release module, the version on the configuration web page will increase with the firmware version, as these have to been updated at the same time.

Here is the table list the versions in following figure:

4.2.6.16 Firmware Version

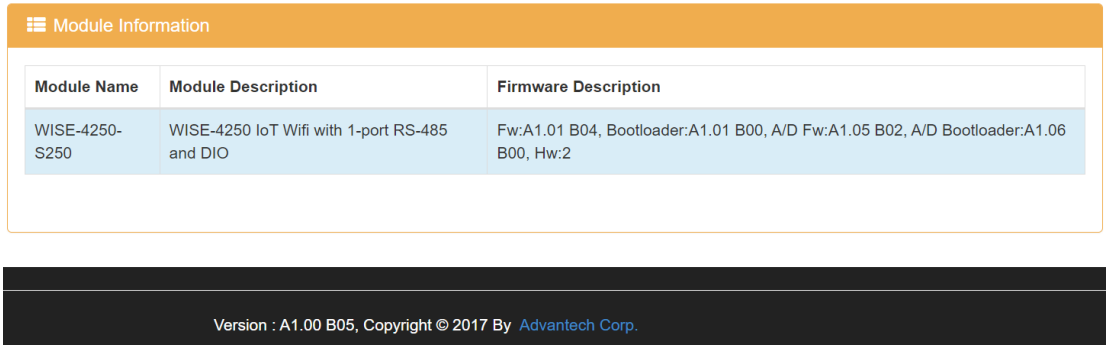


Table 4.7: Firmware Version	
Type	**Version**
Firmware	A1.01 B04
Bootloader	A1.06 B00
Web Page	A1.00 B05

4.2.6.17 System Configuration File

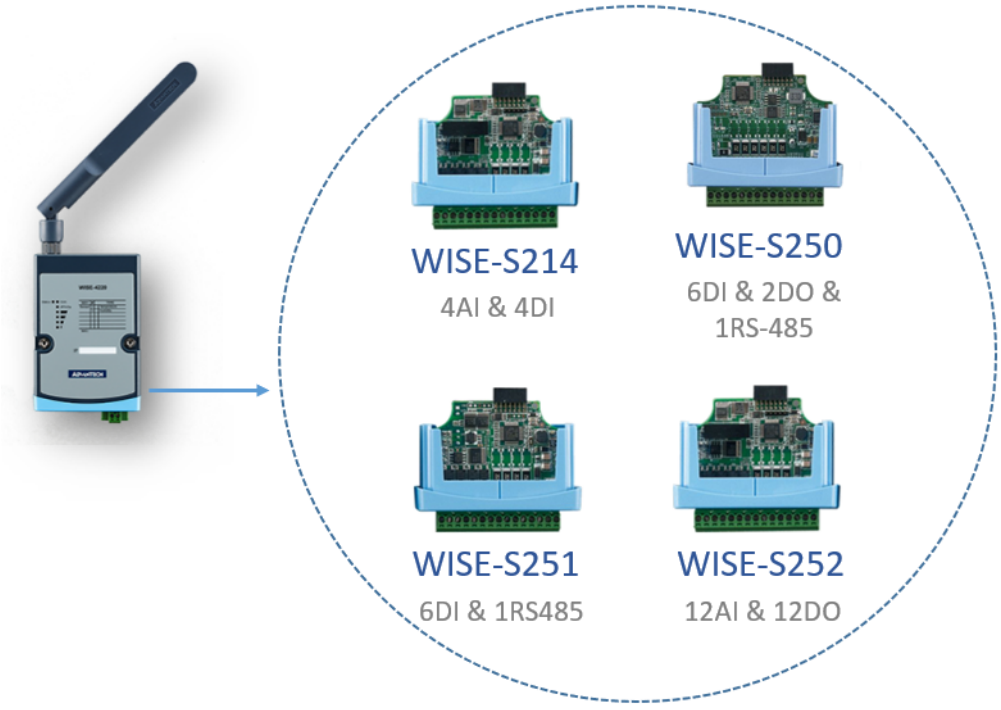
This section demonstrates how to update or download the configuration file from WISE modules. The following items will be saved in the configuration file:

Table 4.8: System Configuration File	
Configuration	Information, Wireless, Network App, Time & Data, SNTP, Modbus, General Cloud, and Account
I/O Status	I/O Configuration
Advanced	Access Control, Data Logger (Data log and Cloud upload)

- Download Configuration File from Module
  Go to the “Firmware” page in system configuration and click the “Download Configuration File” button. The configurations will be saved as a file.
- Update Configuration File to Module
  Go to the “Firmware” page in system configuration and click the icon to select the configuration file you want to upload from the local folder. Before uploading the configuration file to the module, select whether or not to apply the IP settings to the WISE module.

4.2.6.18 IO Status

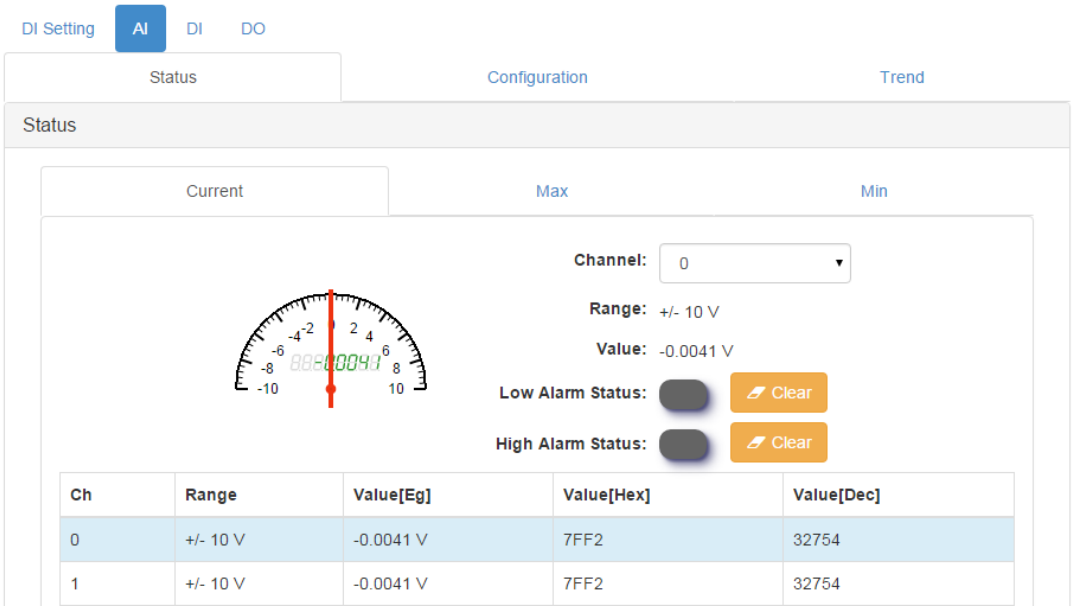
- **IO Status (AI/DI/DO/RS-485)**

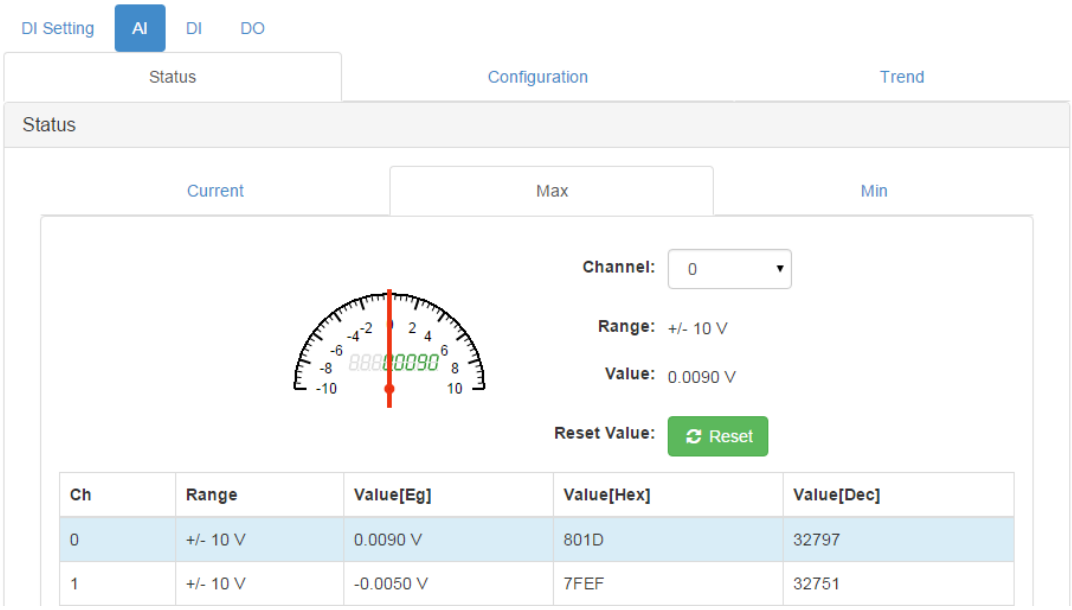


– AI

For the AI channel, the Current/Max/Min status will be shown in the status page, which includes the input range. The current status shows the latest AI value, and also the input range. The average value, which will be introduced in following pages, show the average value of selected channels.

The Max/Min status shows historical maximum or minimum value, you can reset the value by pressing “Reset”.



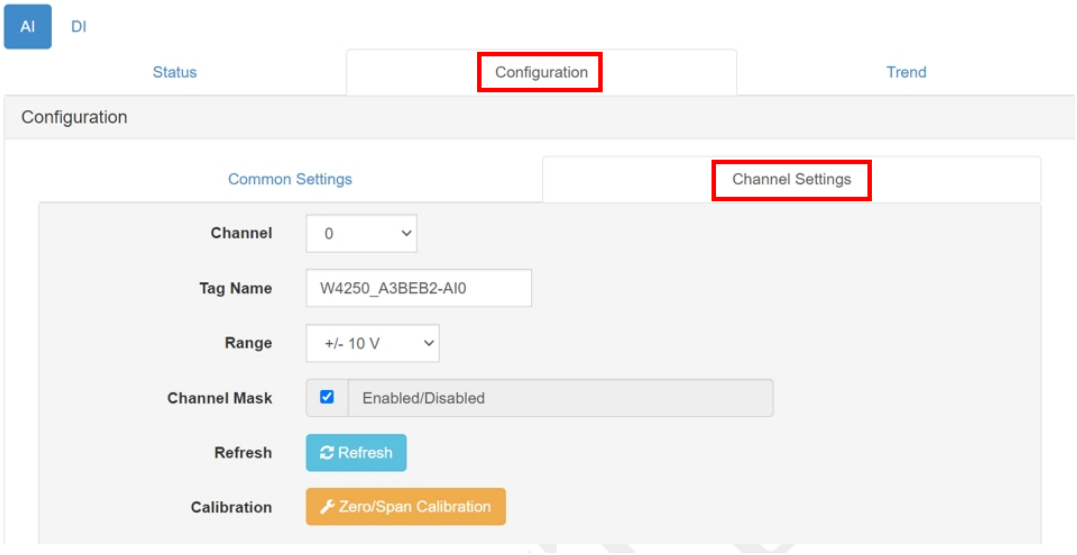


Input Range

For the AI channel which supports more than input range, you can configure the channel setting in “AI/Configuration/Channel Settings”.

Channel Mask

You can disable the AI channel to increase the sampling rate of other enabled channels in “Channel Mask”.



■ Scaling Function

There are two types of scaling function for AI channels:

1. **Input Signal Scaling** - Scaling the Input Range

This is for scaling the analog input range within the configured input range, so that the Modbus value can fit the entire range. For example, Ch0 of WISE-4012 had been configured with an input range of 0~5V, but the full range of signal from sensors is 1~5V. If users do not use the Input Signal Scaling function, the 1V will be presented as 13107 in Modbus, 5V will be presented as 65535. But after Input Signal Scaling, 1V will be presented as 0 in Modbus, and 5V will still be presented as 65535.

If you would like to apply the function in the previous example, enter 1 in “Low Scaling Value”, and enter 5 in “High Scaling Value”.

Note! *The function only increases the resolution of Modbus data, but the accuracy still depends on the original input range before scaling. Furthermore, in the previous example, if the 0~10V input range been used for scaling to 1~5V, it may have a lower accuracy compared to using 0~5V.*

For values which are going to be configured for scaling the input range, note that the “Low Scaling Value” should be lower than “High Scaling Value”.

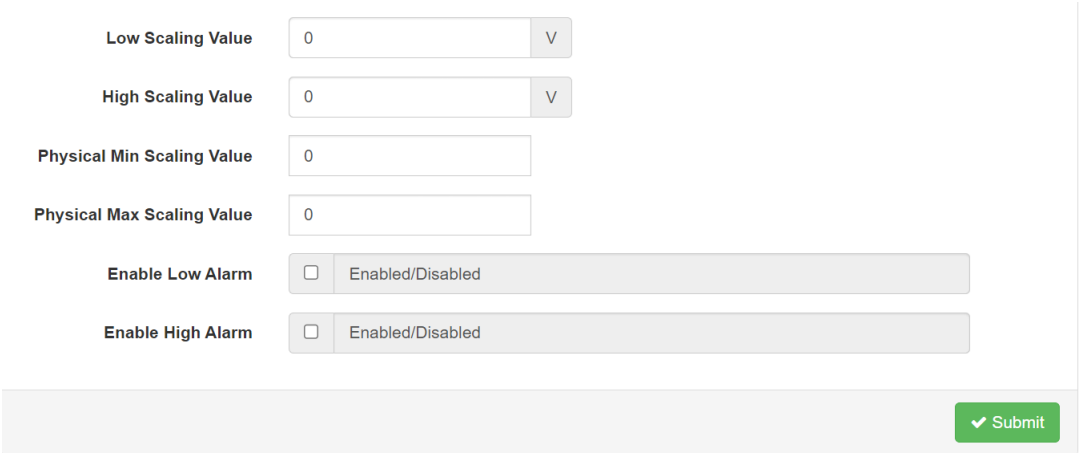
2. **Physical Value Scaling** - Scaling the Output Data

Further to scaling the input range of the analog input channel, the output data can also be scaled. After the function has been applied, it would be easier to read the Modbus value in the engineering unit. For example, a temperature sensor output 0~10 V which shows 0~100°C. It would be better to read 0~100 in Modbus with floating data format.

In the previous case, you can configure the “Physical Min Scaling Value” and “Low Scaling Value” as 0, which shows the input voltage 0 V as physical value 0°C; and configure the “Physical Max Scaling Value” as 100 and “High Scaling Value” as 10, which shows the input voltage 10V as physical value 100 °C.

Note! *The function is helping the data be more readable, but the accuracy still depends on the original input range before scaling and also depends on the sensor’s accuracy.*

For the values which are going to be configured for scaling the output data. For users using RESTful Web API Mapping Unit, can be configured here for further use.



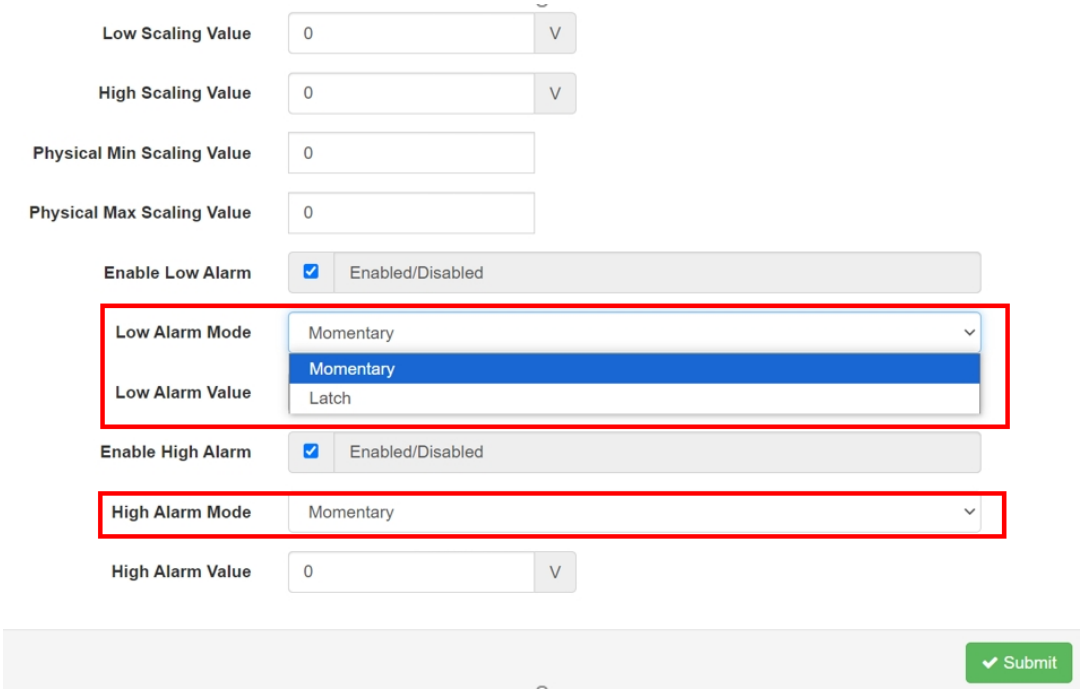
High/Low Alarm

For an AI module with digital or relay output functions featuring a built-in alarm function. When the analog input value is higher than the high alarm value, or lower than the low alarm value, an alarm condition occurs. Then the alarm status will be activated to logic high. The alarm status ia shown in the status page of AI as alarm status LED display, when the alarm condition occurs, the Alarm status LED display will be lit.

The specified digital output channel will generate a logic high value if you build the mapping relationship between alarms and DO channel in the DO mapping area. You can map the DO channel referring to AI Alarm section of DO configuration. The High/Low Alarm status LED in AI status page can be cleared by clicking “Clear”.

This page is for enabling and configuring the alarm. There are two alarm modes:

1. **Latch:** Once the alarm occurs, the alarm status will be activated to logic high level and will keep the value until the alarm is manually cleared. Before the value is cleared, the Alarm status LED will be continuously lit. For an AI module with digital or relay output functions, the specific output channel (chosen in the DO AI Alarm configuration page) will continuously generate logic high value. You can clear the alarm by clicking the “Clear” button in the AI status page.
2. **Momentary:** The alarm status will dynamically change depending on the alarm condition. If the alarm occurs, the alarm status will be logic high. If the alarm condition disappears, the alarm status will be logic low. So not only will the Alarm status LED be lit, in the web page the specific digital output channel value will change depending on the alarm condition.
   After you choose the alarm mode for high alarm or low alarm, you can define the high alarm value or low alarm value by entering the value in Alarm limit text box.



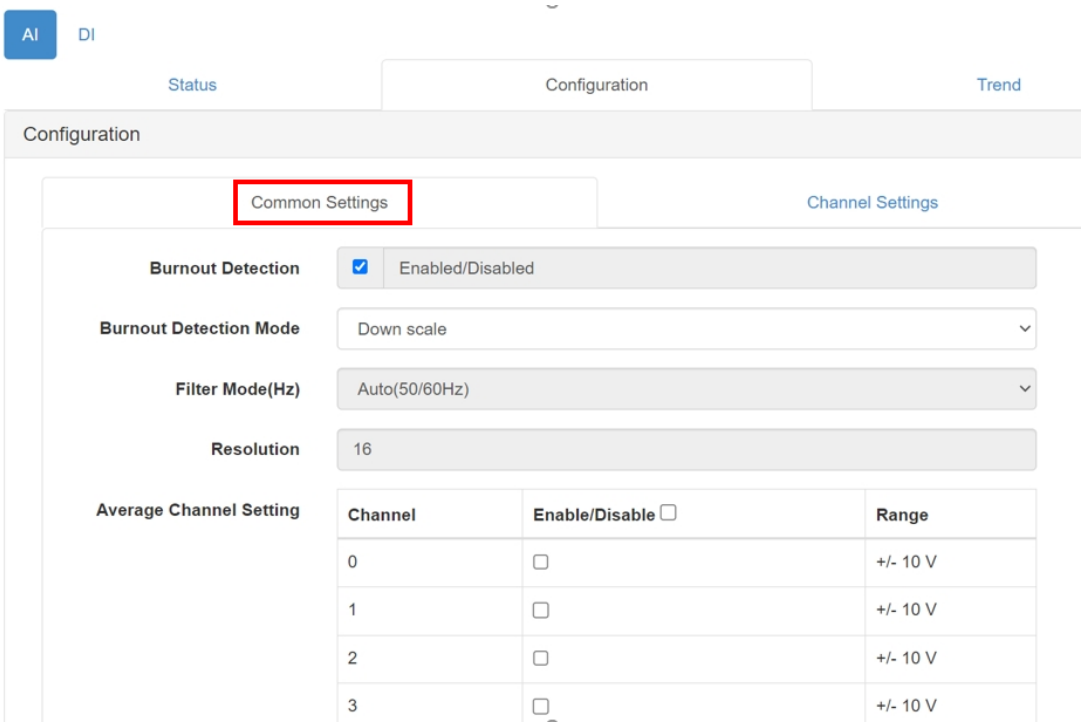
■ Burnout Detection

The Burnout Detection function, or open-wired function, is designed for 4~20mA input range or temperature input range. For the WISE-S214, the burnout signal is activated when the current is less than 3mA. The Modbus flag indicates that the wire of the sensor connected to the channel has burned out. You can also check the Modbus address of AI Channel Status for detail.

When a burnout situation had been detected, the AI value can be shown in “Up scale” which is FFFF(HEX), or “Down scale” which is 0. You can configure this in “Burnout Detection Mode”.

■ Sampling Rate

For models which support more than one sampling rate, you can configure the sampling rate here. For low sampling rate mode, the AI channel would have better noise rejection ability. For the high sampling rate mode, the noise will allow easier coupling to the signal.



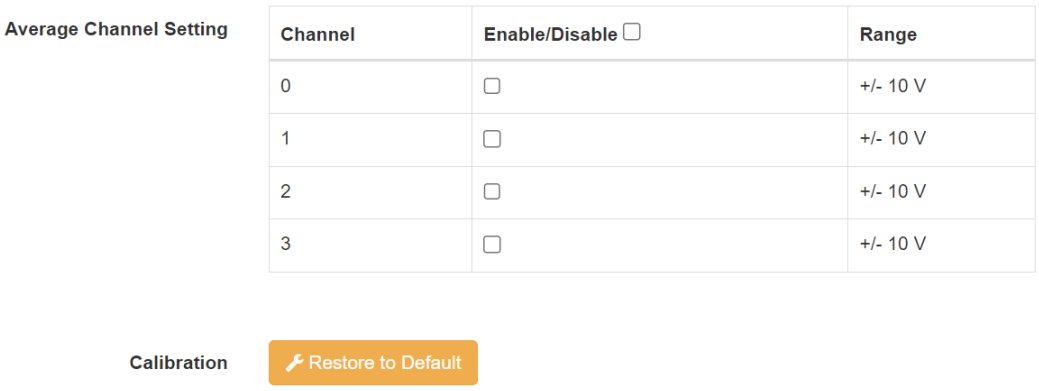
Average Channel Setting

To reduce the data amount, some users don't need the detailed value of each channel but the average value of the selected channel. When the channel is enabled, the values will be averaged in 16-bit integer data, and can be shown or read as another channel.

Calibration

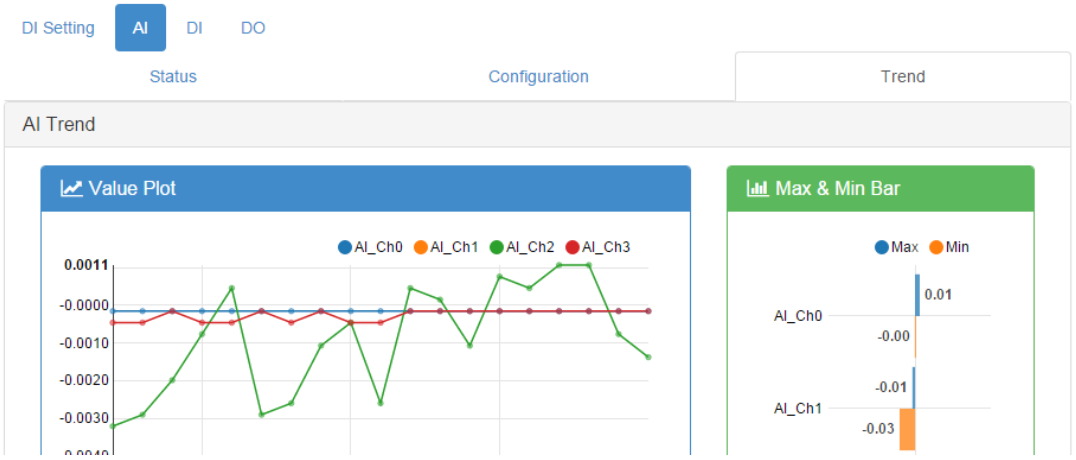
WISE analog input modules support internal reference calibration function, before using the calibration function, you can also try to reset the module to the default factory settings for troubleshooting, or if the calibration process had not succeeded, you can reset the module to the default factory calibration parameters.

Click "**Calibration**" and follow the instructions to calibrate the AI channels.



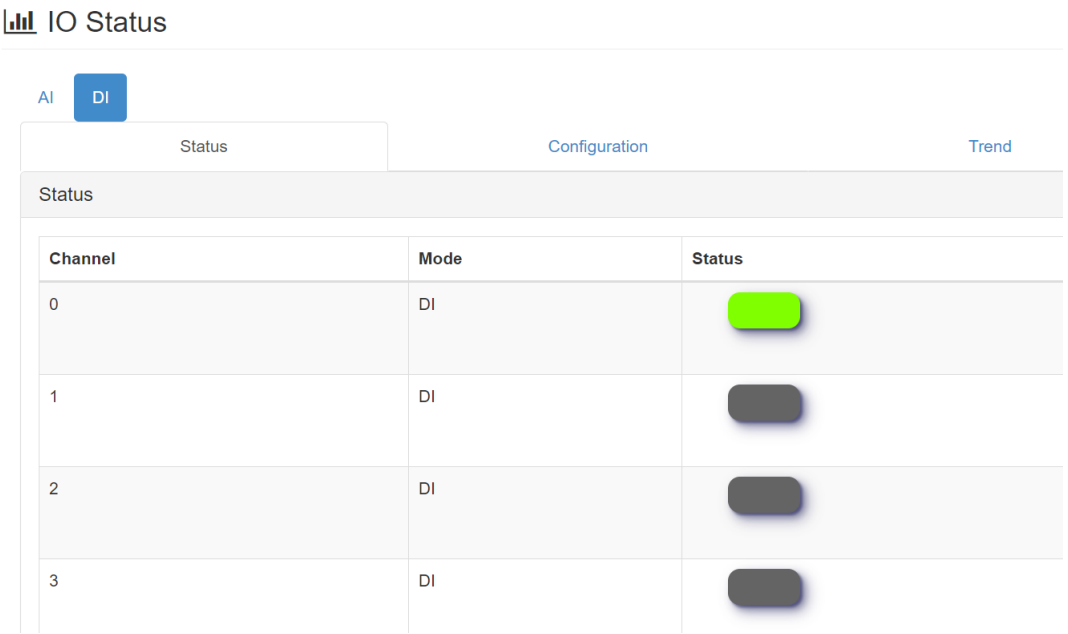
AI Trend

Here you can instantly check the analog input trend of each channel. The function is useful when testing the connection and variety between WISE and sensors.



– DI

You can see the value of all digital input channels by the related LED display in this page. When DI status is a logical high, the LED will be green. If the status is a logical low, the LED will be grey.

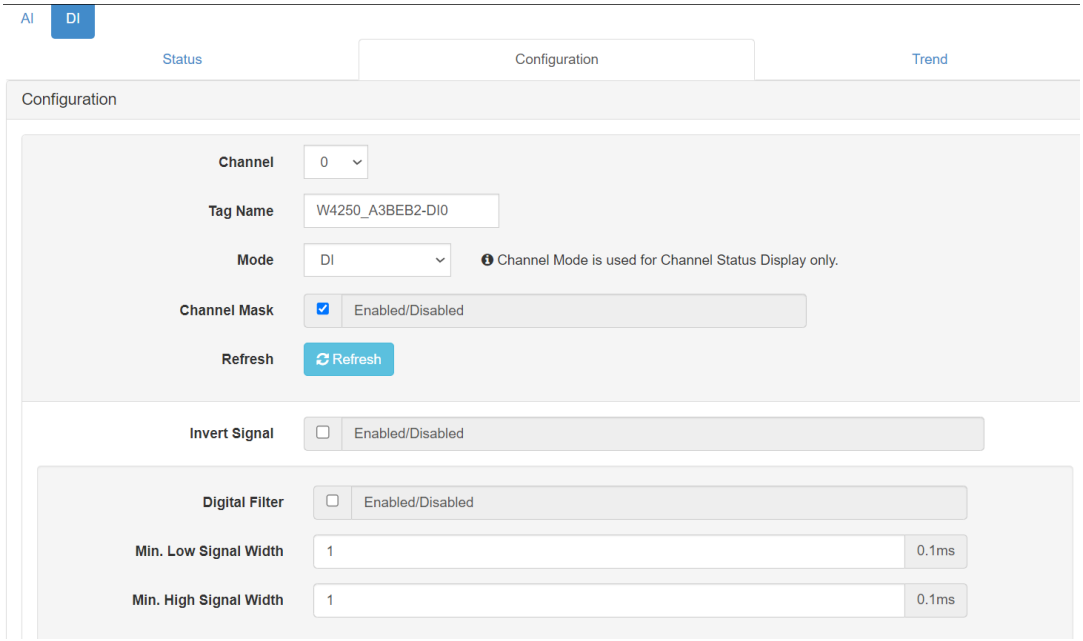


Invert Signal

WISE digital input channels support Invert DI Status function, there will be an Invert Signal check box in the configuration page. Click the check box to enable/disable the function.

Digital Filter

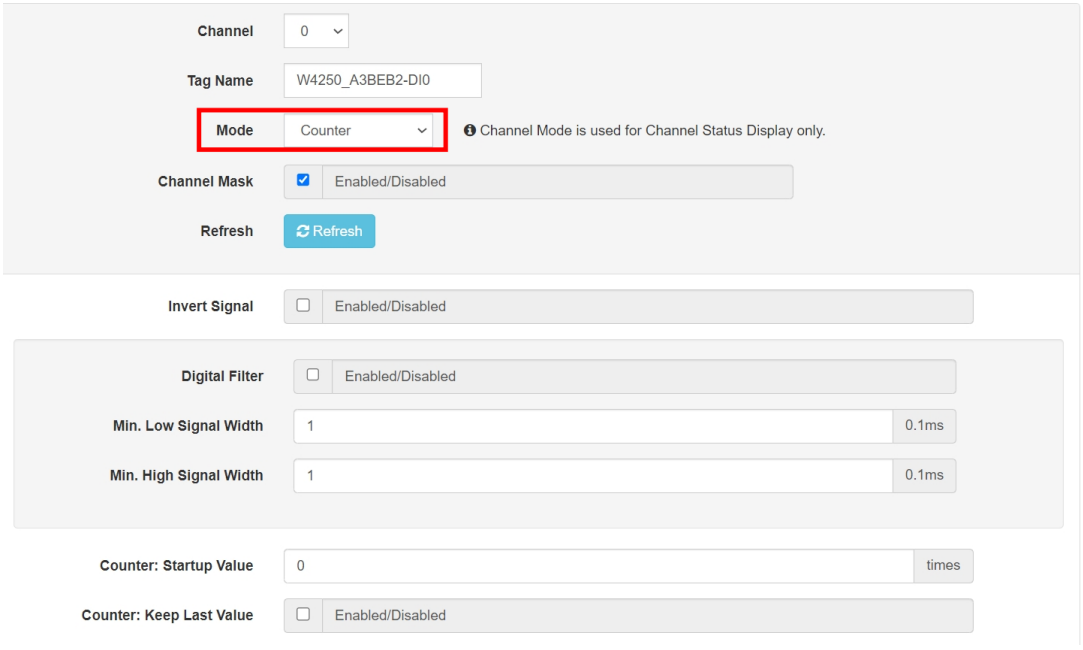
Digital input channels support digital filters, these can be enabled or disabled by clicking the **Digital Filter** check box. If you enable the filter, you can define the minimum acceptable signal width by the **Min. Low Signal Width** and M**in. High Signal Width** text box. (Unit: 0.1ms) The high frequency noise will be removed by this filter.



Counter Mode

When you choose Counter mode, one counter will count the pulse number of the digital signal from the selected channel, and then record the count number in the register. In the DI Status page, the current count value of the selected channel is displayed by the Counter value text box. Start or stop the counter by switching the **Start/Stop** switch next to the Counter value. Reset the counter (the value in the register will be initialized to the startup value, default to be zero) by clicking the **Reset** button. Preset the **Startup Value** in the text box. When you reset the counter value, either by the reset button in the status page or by a command, the value will be reset to the Startup Value. The default value of the **Startup Value** is zero.

Like the DI mode, you can enable/disable the **Invert Signal** function and **Digital Filter** in the configuration page. The operation is the same. If you enable **Keep Last Value**, when the digital module been powered off, the last counter value will be kept in the register. When the module powers on, the counter will continuously count from that value. Without this function, when the module powers off, the counter will reset and the count value in the register will be zero.



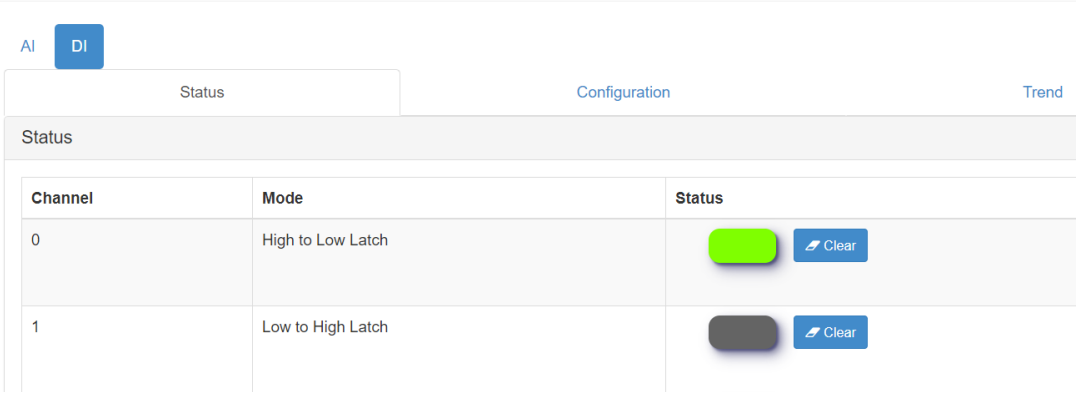
■ Low to High Latch

When you choose **Low to High Latch** mode, once the digital input channel detects logic level changes from low to high, the logic status will be kept as logic high. The logic status will remain the logic high, until you clear the latch manually. The logic status will return to logic low. The logic status can be seen by the Latch status LED display in the DI Status page. Clear the latch by clicking the **Clear** button. Enable/disable the **Invert Status** function in the configuration page.

■ High to Low Latch

When you choose **High to Low Latch** mode, once the digital input channel detects logic level changes from high to low, the logic status will be kept as high. The logic status will remain high, until you clear latch manually. Then the logic status will return to low. The logic status can be seen by the Latch status LED display in the DI Status

page. Clear the latch by clicking the **Clear** button. Enable/disable the **Invert Status** function in the configuration page.

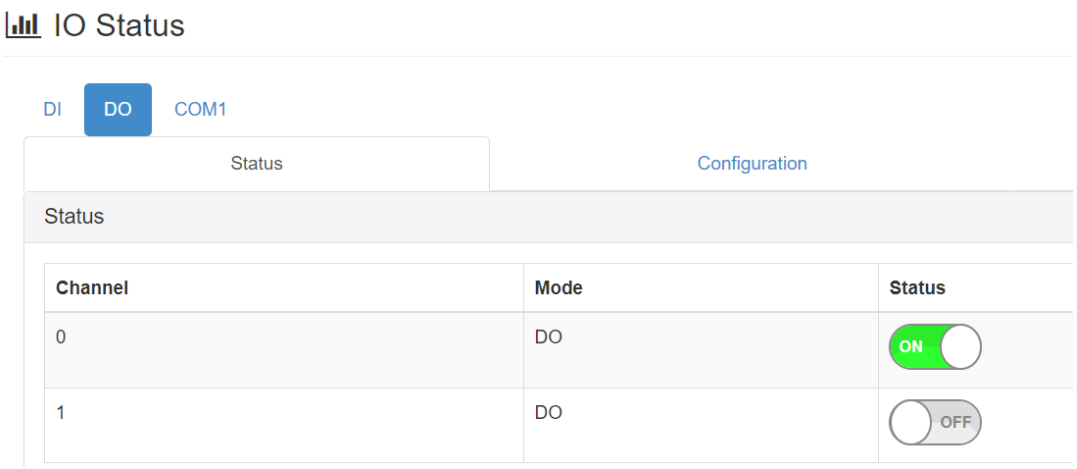


Frequency

For pure DI channels, not including the DI function of the WISE-4012, WISE modules support frequency mode. WISE module will calculate the frequency value of the digital input signal from the selected channel. The frequency value will be displayed in the Frequency value text box in the DI Status page.

– DO

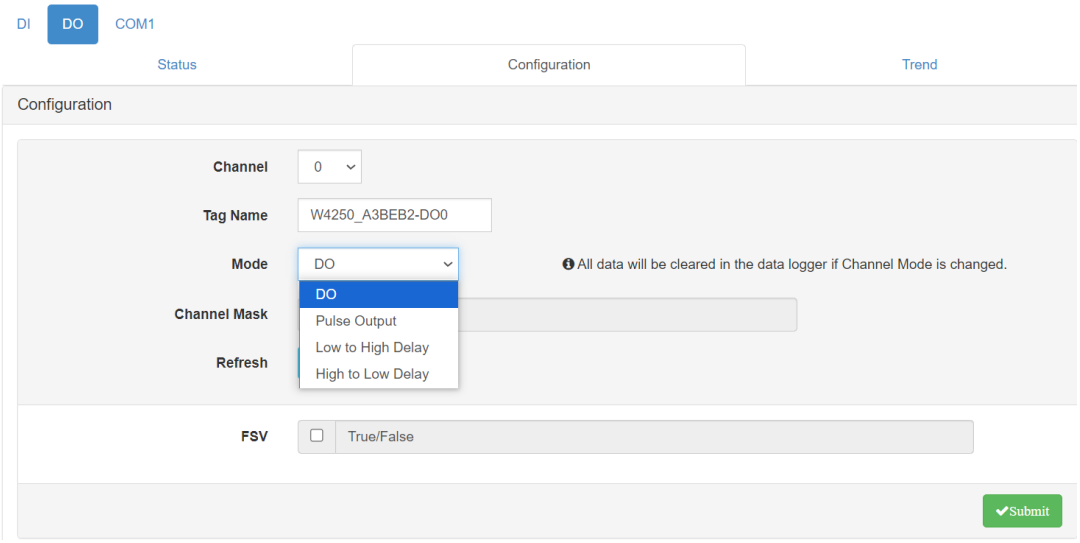
You also can control the values of all digital output channels by the status switch. The color of the switches will display current value of that digital output channel.



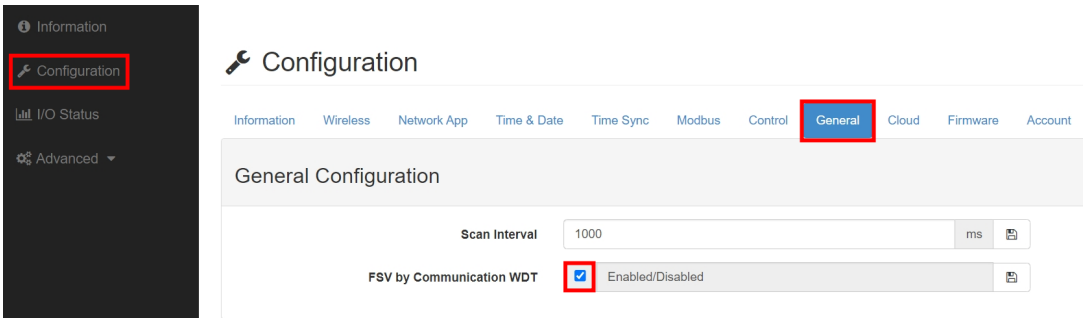
Fail Safe Value (FSV)

When the communication between the host controller and WISE digital modules is broken, the digital output channel can generate a predefined value (this value is called the fail safe value). If the FSV checkbox is checked, the module will set the output channel to logic high when WDT times-out. If the FSV checkbox is unchecked, the module will set the output channel to logic low when WDT times-out.

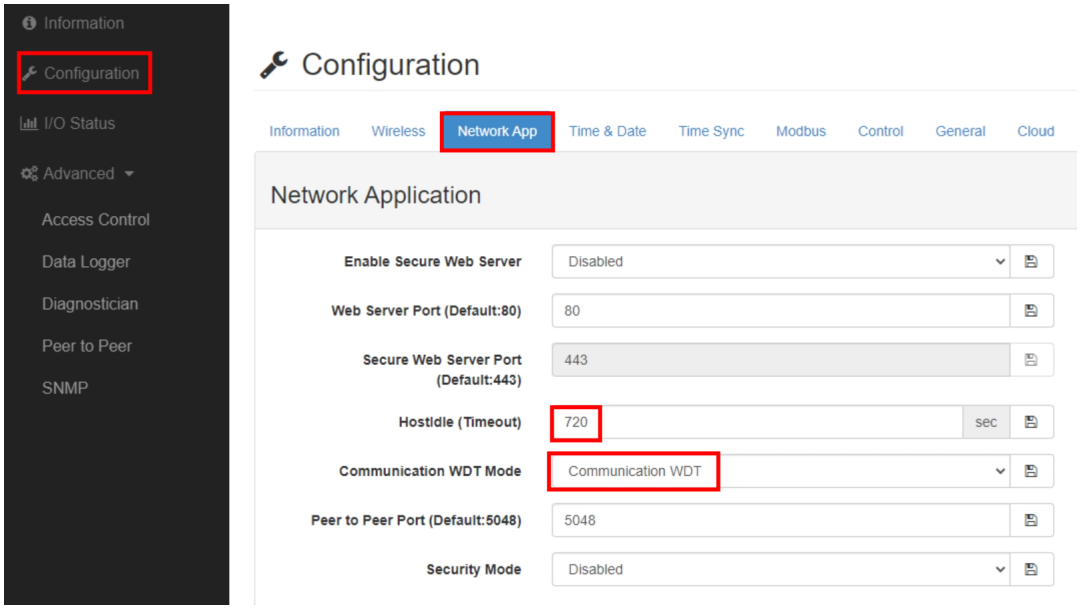
To decide whether to enable the FSV function triggered by communication with the WDT, go to **General Configuration** and enable the FSV function for all the module's output channels.



To decide whether to enable the FSV function triggered by communication with the WDT, go to **General Configuration** and enable the FSV function for all the module's output channels.

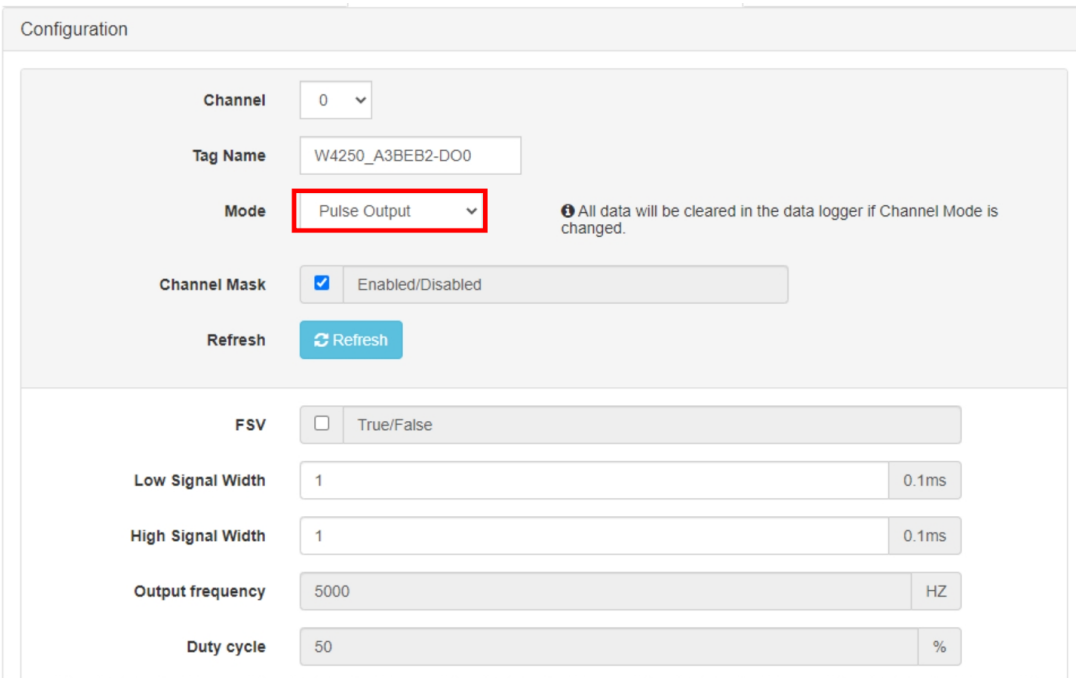


To decide the time period to trigger the communication WDT, go to **Network Application** to enable the **Communication WDT Mode** as **Communication WDT** first, and **0**then configure the **Host Idle (Timeout)**. (Unit: second) The default host idle time is 720 seconds.

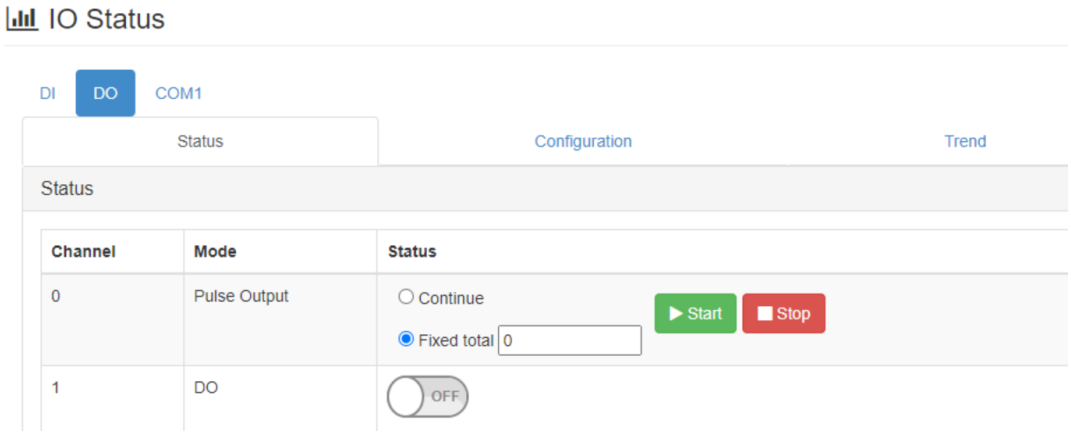


Pulse Output

After you choose the **Pulse Output** mode, the selected digital output channel can generate continuous pulse train or finite pulses. You can define the pulse width by entering into the **Low Signal Width** and **High Signal Width** text box in the configuration page. (Unit: 0.1 ms) The frequency and duty cycle of the pulse output signal will be calculated automatically and displayed by the **Output frequency** and **Duty cycle** text box.



Then choose whether to generate a continuous pulse train or finite pulses by selecting the **Continuous** (for pulse train) or the **Fixed total** (for finite pulses). The text box on the right of the **Fixed total** button is used to define how many pulses you want to generate. After selecting pulse output mode, click the **Start** or **Stop** button to generate or to stop the pulse output.

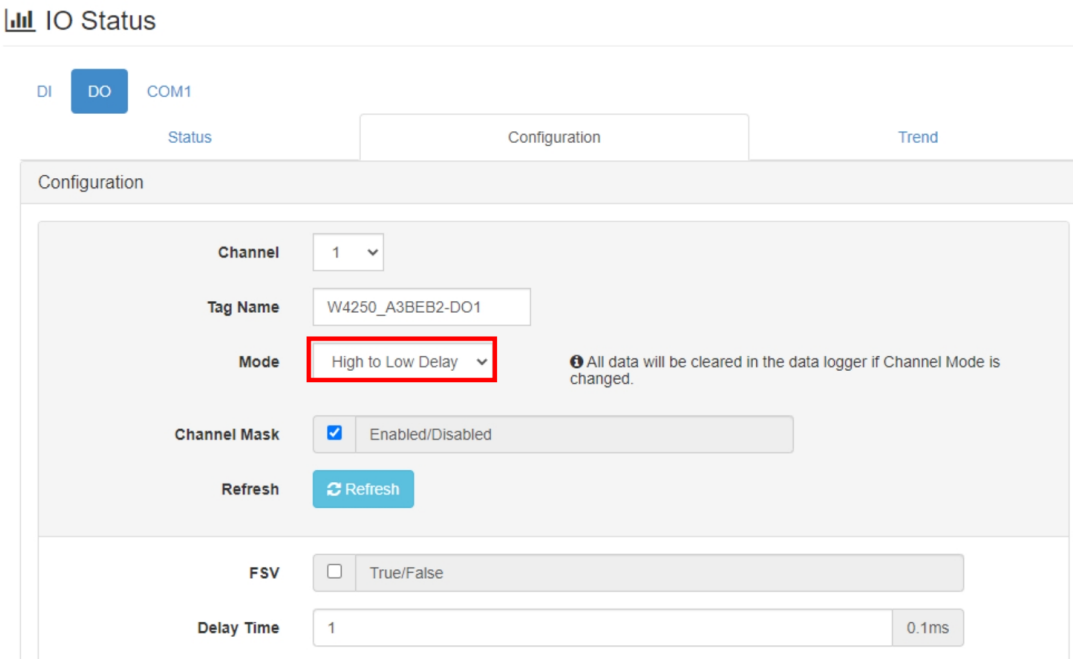


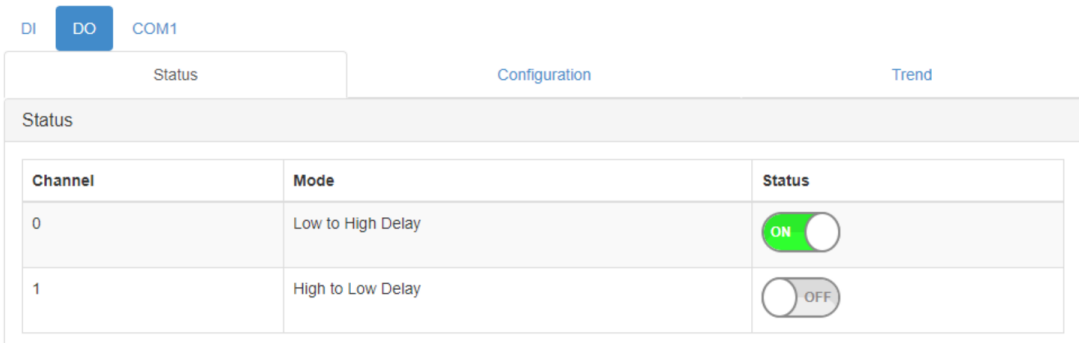
Low to High Delay

Choosing **Low to High Delay** mode, is almost the same as choosing DO mode. The only difference is that there will be certain time delays when the output value changes from logic low to logic high. Define the delay time by entering its value into the **Delay Time** text box in the configuration page. Control the digital output value using the DO button and seeing its current value by the DO status LED display in the DO Status page.

High to Low Delay

Choosing **High to Low Delay** mode, is almost the same as choosing DO mode. The only difference is that there will be certain time delay when the output value changes from logic high to logic low. Define the delay time by entering its value into the **Delay Time** text box in the configuration page. Control the digital output value using the DO button and seeing its current value by the DO status LED display in the DO Status page.





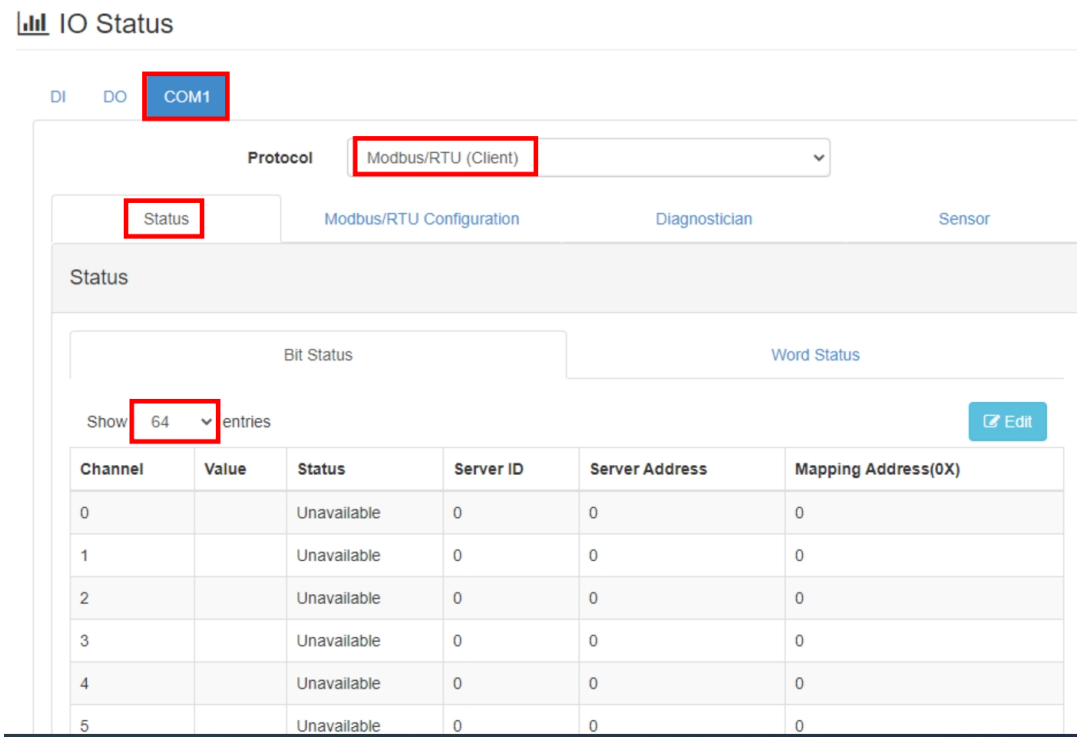
AI Alarm Driven

After the **High/Low Alarm** been configured in AI channel configuration, the alarm status can be mapped in to DO channel. Choose the **High Alarm** or **Low Alarm** in **Trigger Mode** to active the configured DO channel.

- RS-485
  - ◊ Status

WISE-S250 and WISE-S251 have one RS-485 port for Modbus gateway function, thus you can use this port to polling the data from RS-485 Modbus/RTU slave devices.

Go the “COM1” tab to check the status or configure the Modbus Master function of RS-485 port. There can be total 64 addresses of all Modbus slave to be mapped as the I/O of WISE-S250/S251. These 64 addresses can be coils or registers. The coils will be mapped as extension bits of WISE-S250/S251, and the registers will be mapped as extension words of WISE-S250/S251. So, in the “**Status**” tab, you can see the bits or words are shown in individual pages.



Column "**Channel**" indicate the number of bits, there are maximum 64 bits can be shown here, but you may only map less than 64 coils as bits, so the empty bits are invalid. Same as words may also have empty channels.

Column "**Value**" shows the value polling from mapped address.

Column "**Status**" shows the status of each bits or words, if the channel is empty which did not be mapped to Modbus slave address, the status will show "Unavailable".

Column “**Slave ID**” and “**Slave Address**” show where the bit or word from RS-485 Modbus slave device.

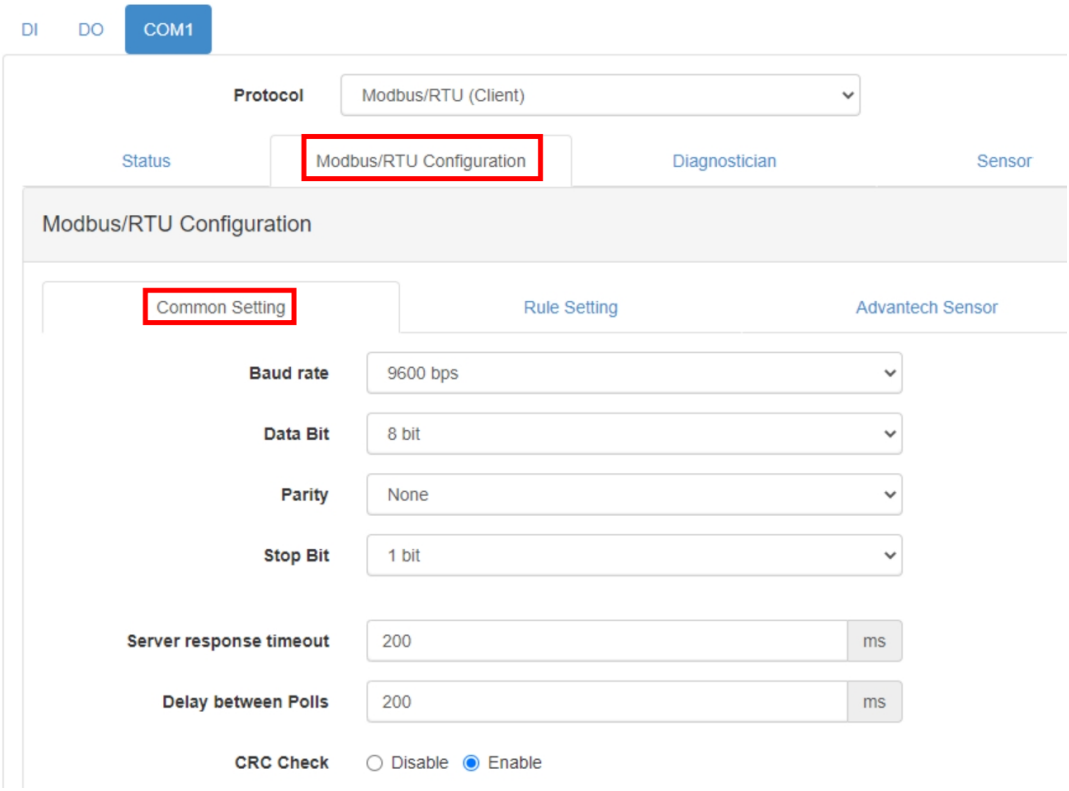
Column "**Mapping Address**" shows the Modbus address of bits or words when WISE-S250/S251 be polled by Modbus/TCP. The default setting of extension bits is from Modbus address 01001 of WISE-S250/S251, and extension words is from Modbus address 41001 of WISE-S250/S251. There are 64 address reserved for extension bits or words for WISE-S250/S251.

For the writable bit or word, you can click “Edit” button to switch to edit mode, change value and click "Apply" to write the Modbus address individually.

■ Modbus/RTU Configuration

In the "Common Setting" Tab, you can configure the parameters of WISE-S250/S251 RS-485 port

- Baud rate: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 bps
- Data Bit: 7, 8
- Stop Bit: 1, 2
- Parity: None, Odd, Even
- Server respond timeout: Here you can configure the time for waiting Modbus slave reply
- Delay between Polls: Here you can configure the delay time between each Modbus instructions
- CRC Check: Here you can enable/disable the CRC check/ignore the CRC error of Modbus



In the "**Rule Setting**" tab, you can configure this Modbus address of end devices you would like to polling.

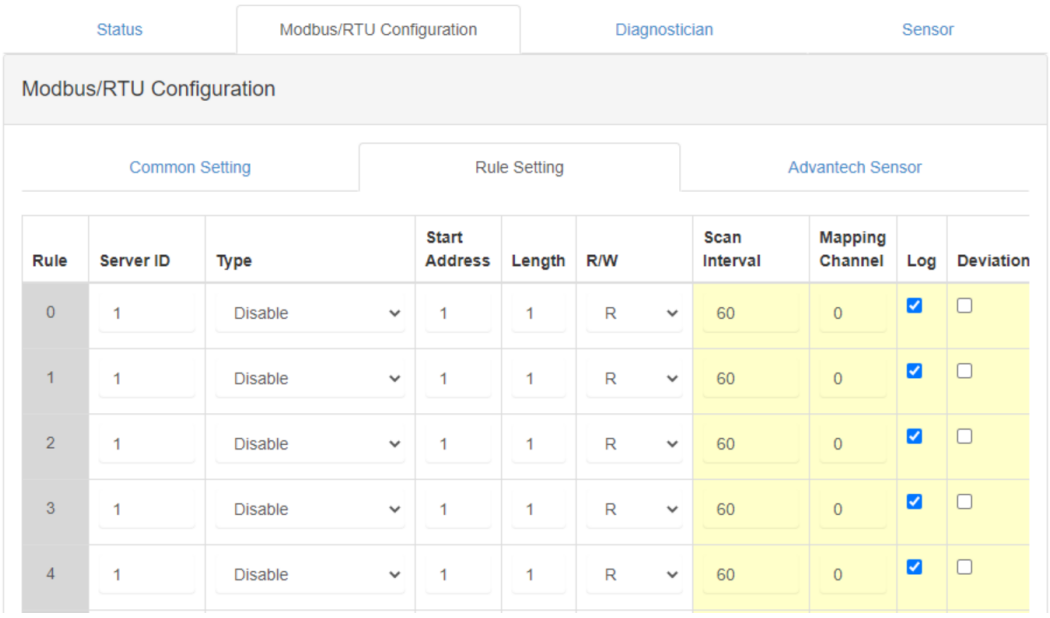
- **Rule:** There are maximum 20 rules that WISE-S250/S251 support. Each rule can be different slave devices, in the other word, it can be maximum 8 devices connected to WISE-S250/S251. Or you can use all rules for polling different address of same slave device.
- **Slave ID:** Different slave devices in same RS-485 has different slave ID, enter the slave address of Modbus devices which connected to WISE-S250/S251 here
- **Type:** We support 4 kinds of Modbus data type, 01 Coil Status (0x), 02 Input Status (1x), 03 Holding Registers (4x), and 04 Input Registers (3x). After you configure one of the types in the rule, then this rule will be enabled, and WISE-S250/S251 will start to polling after the configuration been submitted successfully.
- **Start Address:** Enter the first address number that you are going to polling. The address base is 1, if you are going to polling the first address of Holding Registers, 40001, please enter number 1 here. Don't need to enter the whole address 40001.
- **Length:** Enter the length of the address that you are going to polling in this rule. For example, if you are going to polling 40001~40008, enter the length as 20 here. Please be noted that since WISE-S250/S251 can polling maximum 64 address, the maximum length is 64 addresses, and the total amount of all rules should also less or equal to 64.
- **R/W:** Here you can decide if the address in this rule will be Read or Written or not. For Coil Status and Holding Registers, you can make these addresses read only, or write only to reduce the polling effort.
- **Scan Interval (in milliseconds):** Here decide the scan interval for WISE-S250/S251 to polling Modbus slave devices. WISE-4051 will optimize the scan interval according you setting. However, the read scan interval may also depend on

real case like: Baud rate, slave devices respond time, delay time between polls, etc. Go to the Diagnostician page to check the real respond time for referring the value of scan interval.

- **Mapping Channel:** When the Modbus address of slave devices been configured in each rule, these addresses will also be mapped into WISE-S250/S251. Coils of Modbus slave devices will be mapped as bits for RESTful web service and also be mapped as coils for Modbus address of WISE-4051. Registers of Modbus slave devices will be mapped as words for RESTful web service and also be mapped as registers for Modbus address of WISE-S250/S251. There are 64 continuous channels of bit and another 64 continuous channels of word can be mapped. Please make sure the channels for each rule are not overlapped.
- **Log:** Here you can decide the data been polled from this rule will be logged in data logger or not.
- **Rule Status:** The web configuration interface will check if rule settings have any overlapping or confliction. The enabled rules (enable the rule by configuring "Type") should have green icon so that the "Submit" will been shown for submitting the rules.

Note:

1. After configuring the rules, click "Submit" to apply the rules.
2. After changing the rule configurations, the logged data in data logger will be cleared for organizing new data structure of data logger for new configurations.
3. Place your mouse over the table title to show the tips.

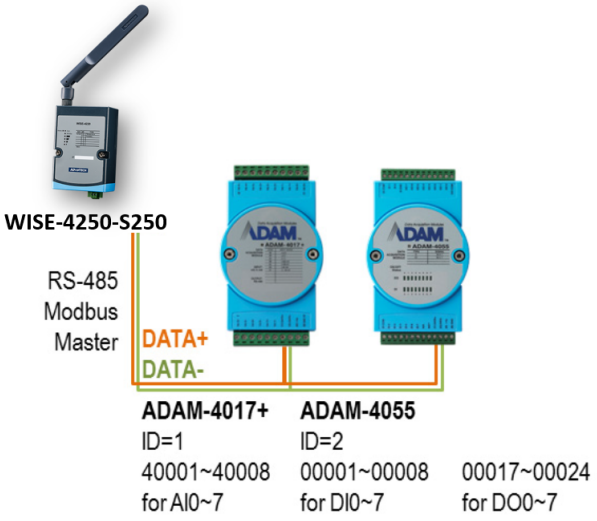


In previous figure we demonstrate how to configure an ADAM-4017+ (or ADAM-4117) which slave ID is 1 and an ADAM-4055 which slave ID is 2 as the Modbus slave devices connected to WISE-4250-S250 +ADAM-4017+ (or ADAM-4117) is an 8-ch analog input Modbus I/O modules, the Modbus address of AI0~AI7 are 40001~40008. In the **Rule Setting** page, we configure the **Slave ID** = 1; **Type** = 03 Holding Registers, since the Modbus address 40001 is start from 4; **Start Address** = 1 and **Length** = 8 for the address 40001~40008; **R/W** = R, since this address is for analog input which is read only; **Scan Interval** =1000ms for polling every second; **Mapping Channel** = 0, to mapping the data from AI0~7 of ADAM-4017+ to channel 0~7 of Word Status; and check the **Log** to log the data from ADAM-4017+.

ADAM-4055 is an 8-ch digital input and 8-ch digital output Modbus I/O modules, the Modbus address of DI0~DI7 are 00001~00008; the Modbus address of DO0~DO7 are 00017~00024. Since the address is not continuously, so we are going to configure digital input as one rule, and digital output as another rule.

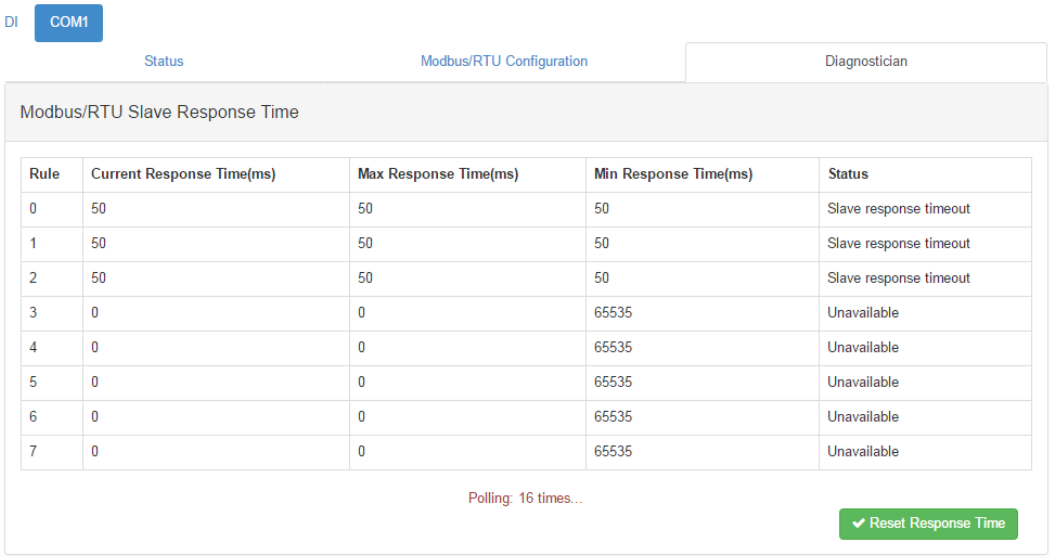
For digital input channels: In the **Rule Setting** page, we configure the **Slave ID** = 2; **Type** = 01 Coil Status, since the Modbus address 00001 is start from 0; **Start Address** = 1 and **Length** = 8 for the address 00001~00008; **R/W** = R, since this address is for digital input which is read only; **Scan Interval** = 1000ms for polling every second; **Mapping Channel** = 0, to mapping the data from DI0~7 of ADAM-4055 to channel 0~7 of Bit Status; and check the **Log** to log the data from ADAM-4055. Please be noted that the Bit Status and Word Status have individual channel number, so the Word Status for ADAM-4017+ and the Bit Status for ADAM-4055 are all start from 0.

For digital output channels: In the **Rule Setting** page, we configure the **Slave ID** = 2; **Type** = 01 Coil Status, since the Modbus address 00017 is start from 0; **Start Address** = 17 and **Length** = 8 for the address 00017~00024; **R/W** = R/W, since this address is for digital output which can be wrote and read, you can also configured as W if you don't want to read back the value; Scan Interval = 1000ms for polling every second; **Mapping Channel** = 8, to mapping the data from DO0~7 of ADAM-4055 to channel 8~15 of Bit Status; and check the Log to **log** the data from ADAM-4055. Please be noted that the channel 0~7 of Bit Status have been occupied by previous rule, so you should assign the channel number from channel 8~31.



■ Modbus Slave Devices Diagnostician

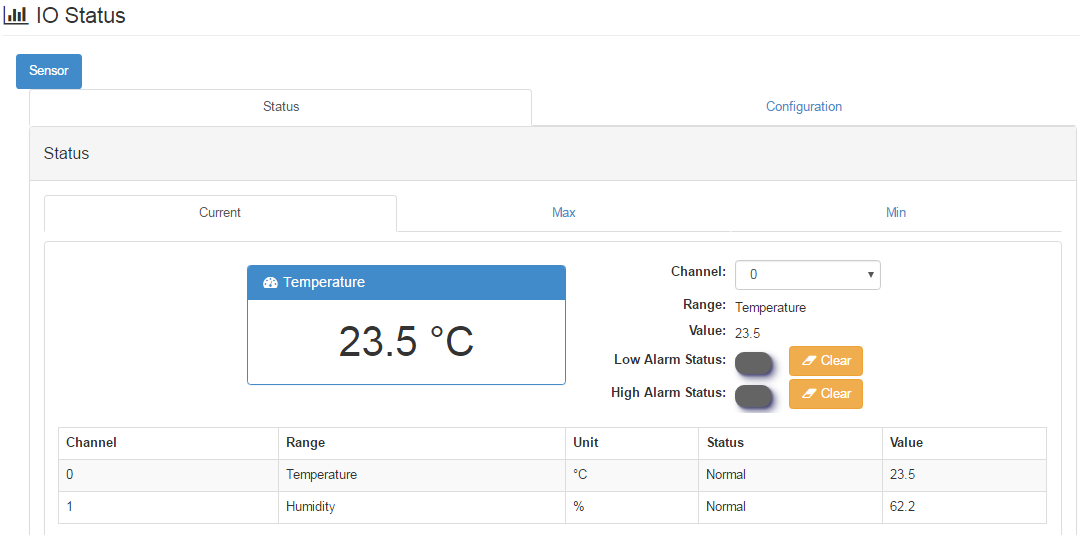
Since different devices will have different responds time, to have better configuration of scan interval, here WISE-S250/S251 provides Diagnostician function for testing the respond time of each rule. You can refer to the respond timeout in this page for configuring the "**Scan Interval**" in "**Rule Setting**" page. You can reset the testing result in this page by clicking "**Reset Response Time**".

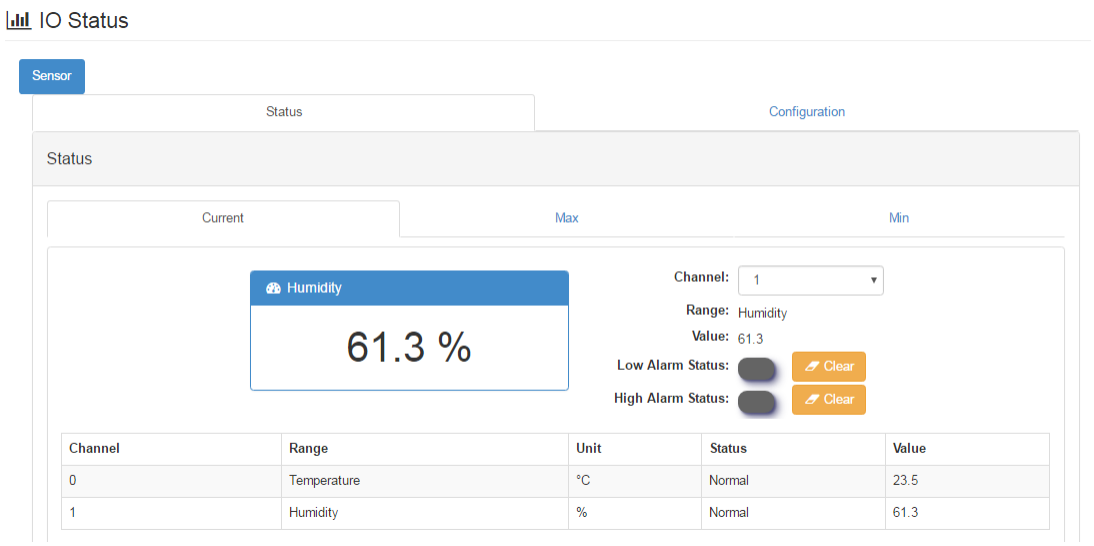


The IO family of WISE-4250 series is composed of various types of IO, including DI, DO AI, RS-485, temperature and humidity or other sensors. You can flexibly select the I/O models which meet your requirements based on the applications and even can contact our sales for I/O customization based on business cooperations.

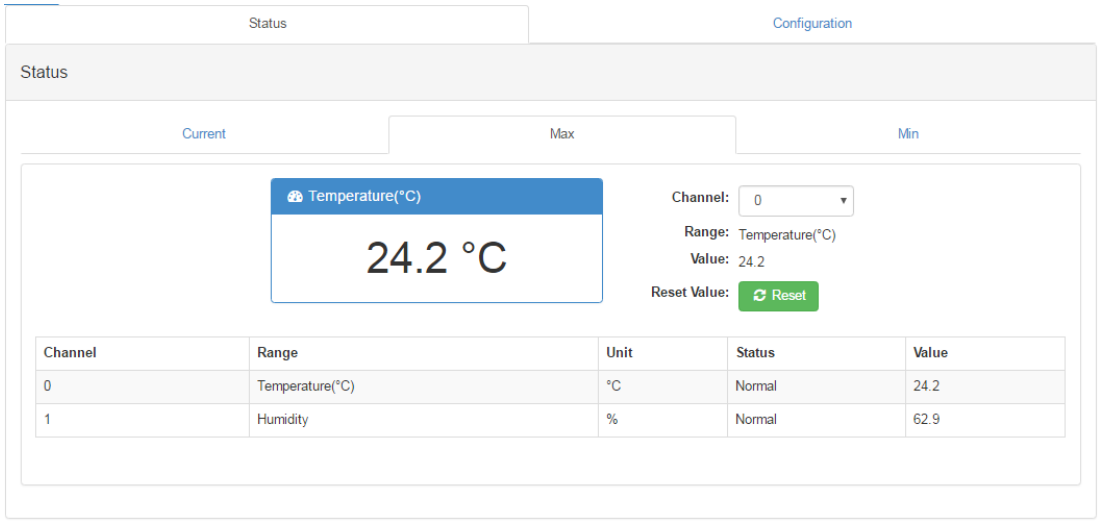
■ WISE-S232 Temperature/Humidity Sensors

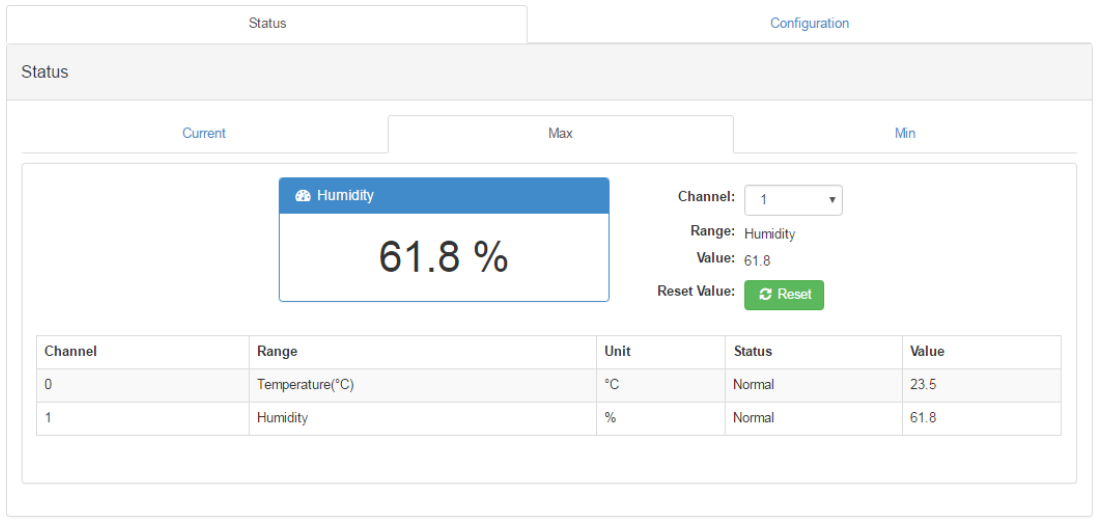
It is equipped with built-in temperature and humidity sensors. As you can see, the IO status section shows measured results. Channel 0 and 1 present the measured temperatures and humidity from the sensors, respectively.





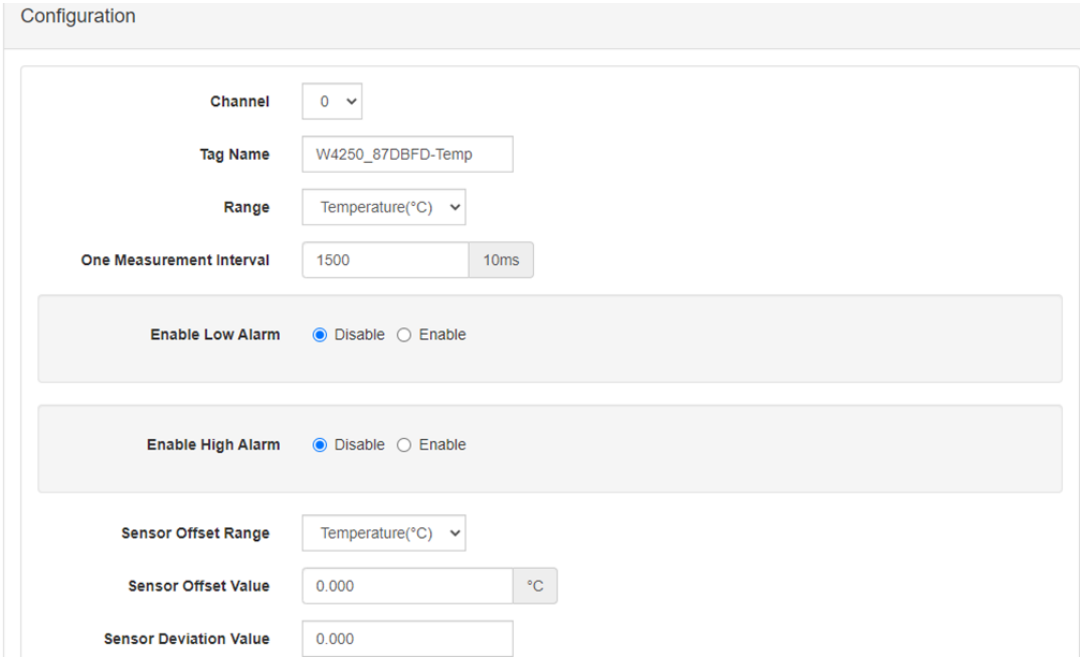
You can also view the maximal and minimal measured results from the historical record by selecting “Max” and “Min” tabs. By clicking “Reset,” you can reset the maximal and minimal records in the module.





You can also change the settings of channel 0 and 1 in the “configuration” tab. The fields that can be changed are shown as follows:

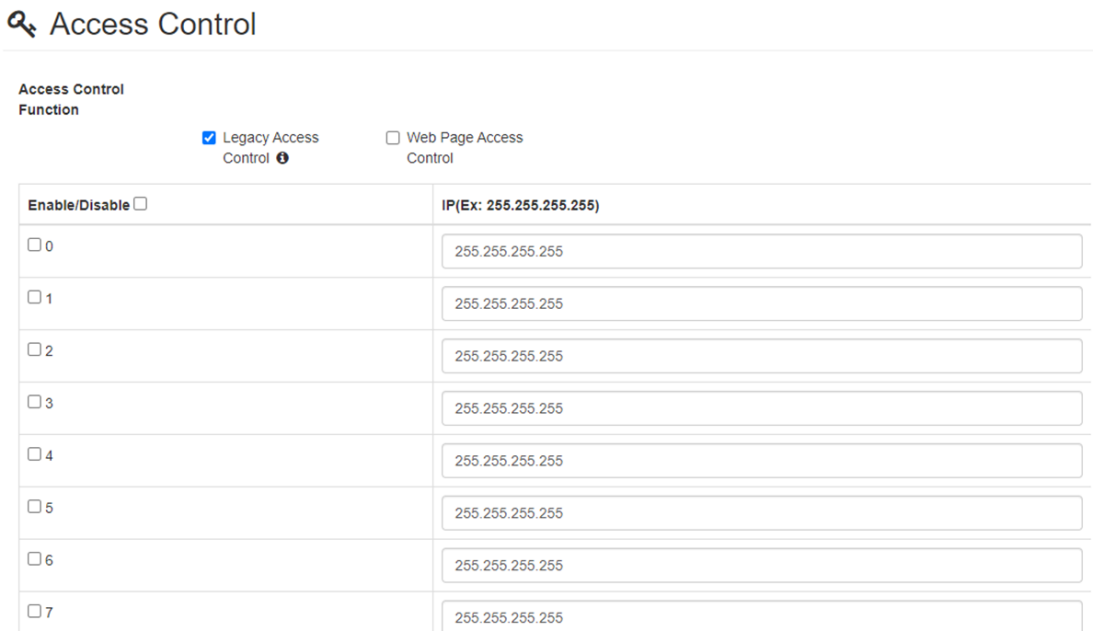
- **Tag Name:** The name of the node
- **Range:** The unit used by each measured factor
- **One Measurement Interval:** The time interval between two measurement events
- **Enable Low Alarm:** If you enable this option, the module will send out an alarm signal when the measured results are below than the lower limit.
- **Enable High Alarm:** If you enable this option, the module will send out alarm signal when the measured results are higher than the lower limit.
- **Sensor Deviation Value:** The difference of measured results of two consecutive events. If the measured difference is higher than the deviation value you set, it will trigger an event.
- **Sensor Offset Range:** The unit used to offset the measured number.
- **Sensor Offset Value:** The delta value used to offset the measured number displayed in the “Status” section.



4.2.7 Advanced Function

4.2.7.1 Access Control for Security

To avoid unauthorized access, you can manage the list of devices that can access the WISE-4250 module by using the “Access Control” section.Please enable one of the rows and enter the IP address or MAC address of the devices that are allowed to access the WISE-4250 module.There're Legacy Access Control and Web Page Access Control can be chose.



4.2.7.2 Data Logger

The WISE-4250 series supports data log functions. I/O status can be logged in the module and queried from the module.

- **Time & Date/Time Sync**

Before you start the log function, please make sure that the RTC time inside the WISE module is correct. Standard WISE modules, excluding WISE-4012E, come with an RTC battery. Once the RTC time has been configured, you don't need to synchronize the time with SNTP server. The time will be kept in the RTC with a battery. For the WISE-4012E without an RTC battery, you need to synchronize the time with SNTP server.

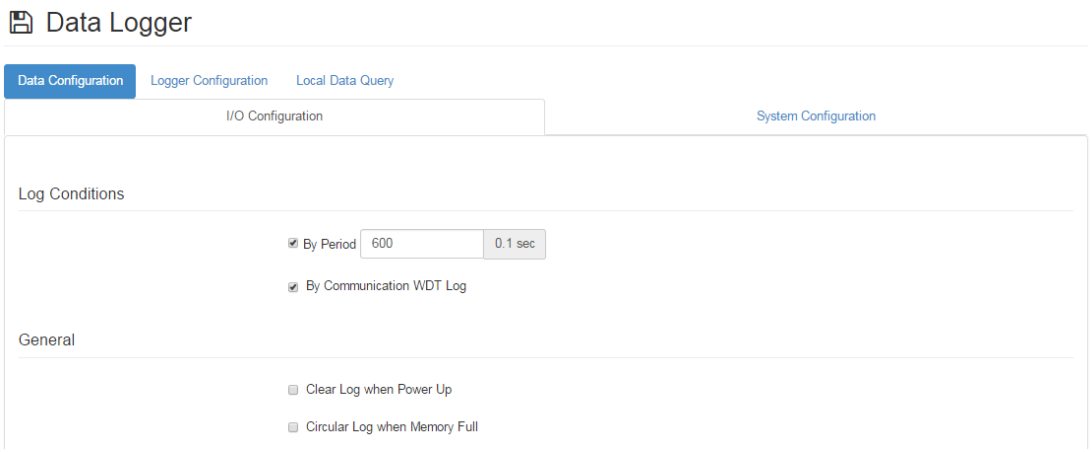
- **Data Configuration**
  - I/O Data

You can use the “Log Conditions” section in “Data Configuration” to choose the method that will be used to log data. If you check “By Period” box, it will enable periodic logging, and the log period can be assigned in the following field. Note that the unit of this field is 0.1 second, meaning that the status of I/O will be logged every minute if you enter “600” here. If you check the box “By Communication WDT”, once the condition of the WDT is met, the status of the I/O will be logged.

For the analog input channel, data can be logged by the AI Deviation Rate (Dividing difference between present sample value and previous sample value by the total range value). Here you can enter the percentage of the deviation rate to be the criteria for triggering the logger.

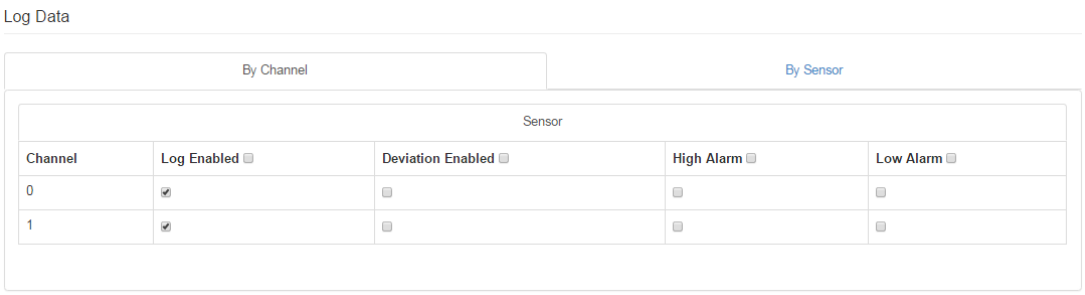
All the data can be kept even if the module is powered off; however, you can clear all data in the logger when powering up WISE module by checking the “Clear Log when

Power Up" box. When maximum memory capacity is achieved, the logger will stop logging by default. You can check the "Circular Log when Memory Full" box to overwrite the memory.

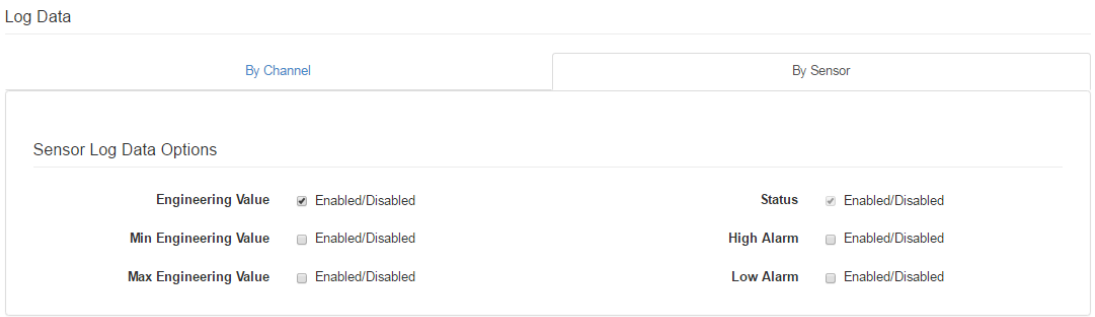


The "By Channel" tab is used to determine which kind of status will be logged and whether the change of the status will be logged or not. Note that the log memory will be cleared if any parameter is changed in the "By Channel" and "By Sensor."

For sensor channel, check the "Log Enabled" box to log the status of checked channel periodically. After you check "Log Enabled" box, you can check the "Deviation Enabled" box if you want to log data when the deviation matches the criteria set in "Sensor Deviation Value" field of the "IO Status" section. You can also choose to log the events that trigger high and low alarms by checking "**High Alarm**" and "**Low Alarm**" boxes.



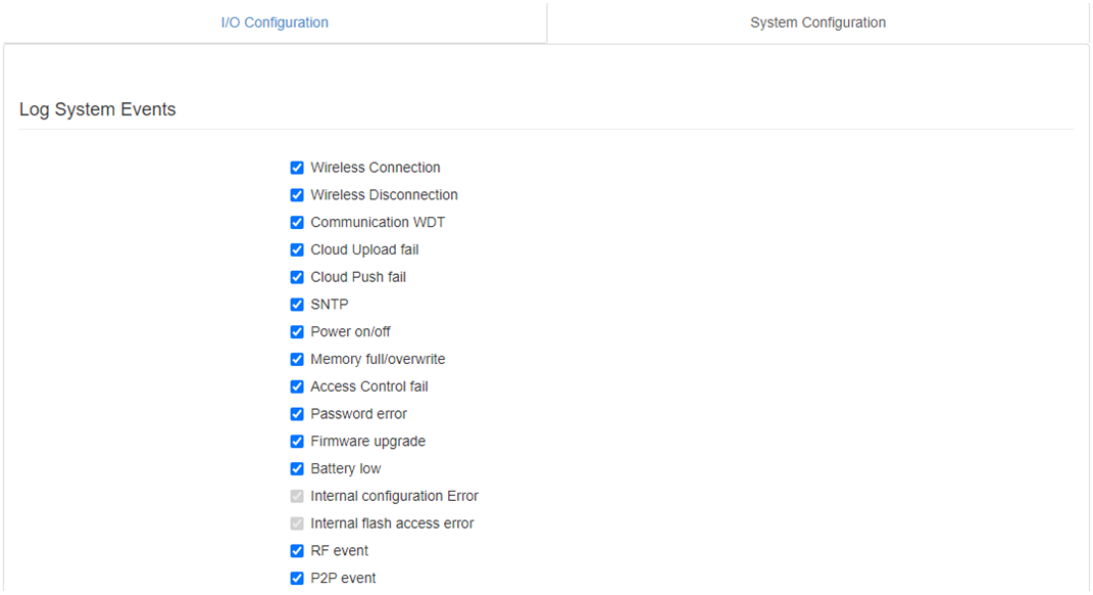
The "By Sensor" tab is used to decide what kinds of data will be logged. Note that the log memory will be cleared once any parameter changed in the "By Channel" and "By Sensor" tabs.



4.2.7.3 System Data

The WISE data logger function not only logs the I/O status, but also logs system events for module diagnostics or troubleshooting. You can choose what kind of system events you would like to log here.

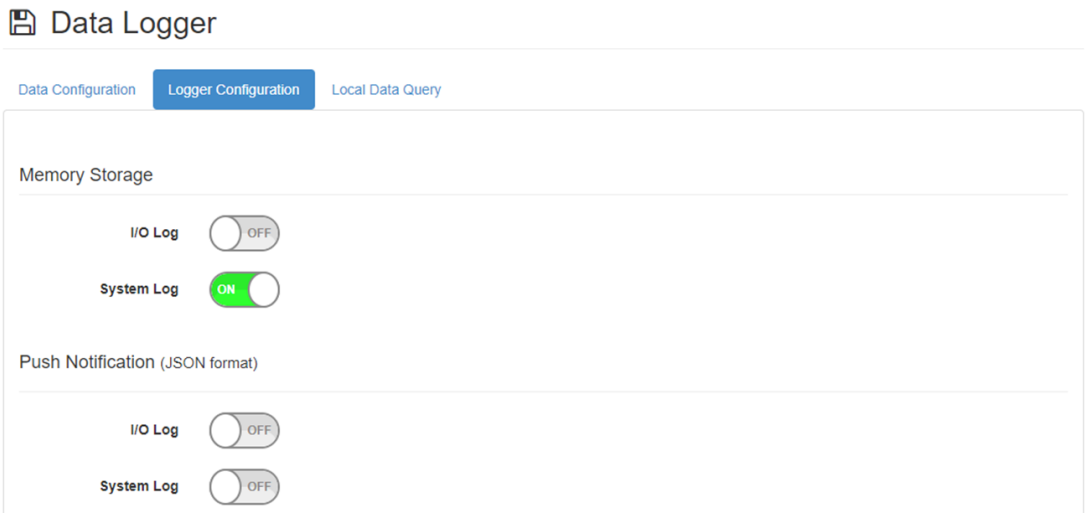
– Enter to the System Configuration setting of Data Logger page.
(Advanced → Data Logger → Data Configuration → System Configuration)
– Check to select the system events to log.



Logger Configuration

In the previous page, you can configure which data is logged. In this page you can enable the local memory storage function. There are separate switches to enable the function of logging I/O data and system data. You can turn ON the switches to enable the function. It also support JSON format Push Notification.

– Enter to the Logger Configuration page.
(Advanced → Data Logger → Logger Configuration)
– Enable the System Log switch to record the system log to the memory storage of WISE modules



Local Data Query

The logged data can be queried from the WISE module. Due to the limitation of MCU-based WISE modules, the file will be saved in a *.json file. You can visit https://json-csv.com/ to convert the data from *.json to *.csv.

Before querying the logged data, you can configure the format of the file. You can determine whether the data comes with a UUID or MAC ID, and choose the type of time stamp. For the latest version of WISE module, which supports Local Date and

Time (GMT), the time stamp will be reported in the format like this: "2015-08-27T15:20:29+08:00", whereas the time stamp will look like: “1440660089,” if it supports Coordinated Universal Time (UTC).

After selecting the data format, you can query the amount of data or data within a certain period by selecting “Amount of Latest Data” or “Time Filter” modes, respectively. However, if the amount of data is not too large, you can also choose “No Filter Enabled” to query all the data.

Now you can click Query to query the data from local memory. Then the data will be shown in a chart and table. Click “Save” to save data from the WISE module into a *.json file, or you can click “Clear” to clear all data in local memory.

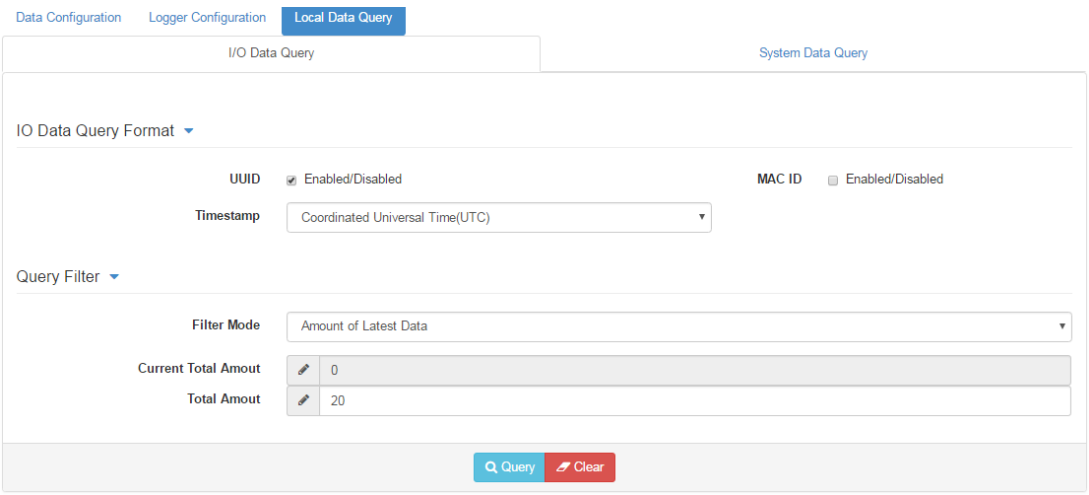
– Enter to the System Data Query setting of Data Logger page.
  (Advanced → Data Logger → Local Data Query → System Data Query)
– Select the System Data Query Format you want to display in system log.

UUID: The default UUID is the combination of model name and the MAC address.

MAC ID: The MAC ID of WISE module.

Timestamp: There are two types of timestamp to choose, GMT and UTC.

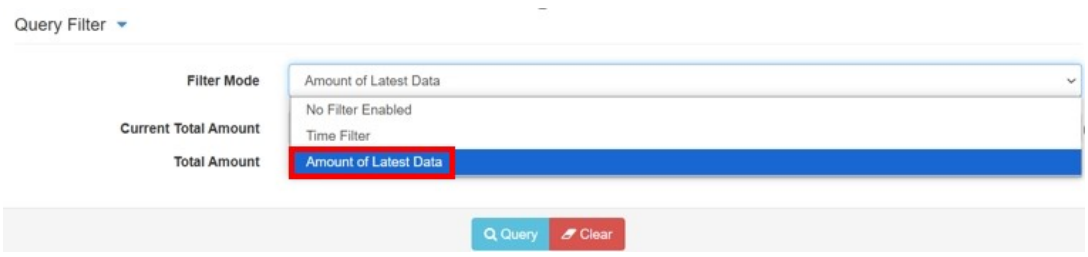
In this case, we enable the UUID and select the GMT Timestamp as our system data query format.



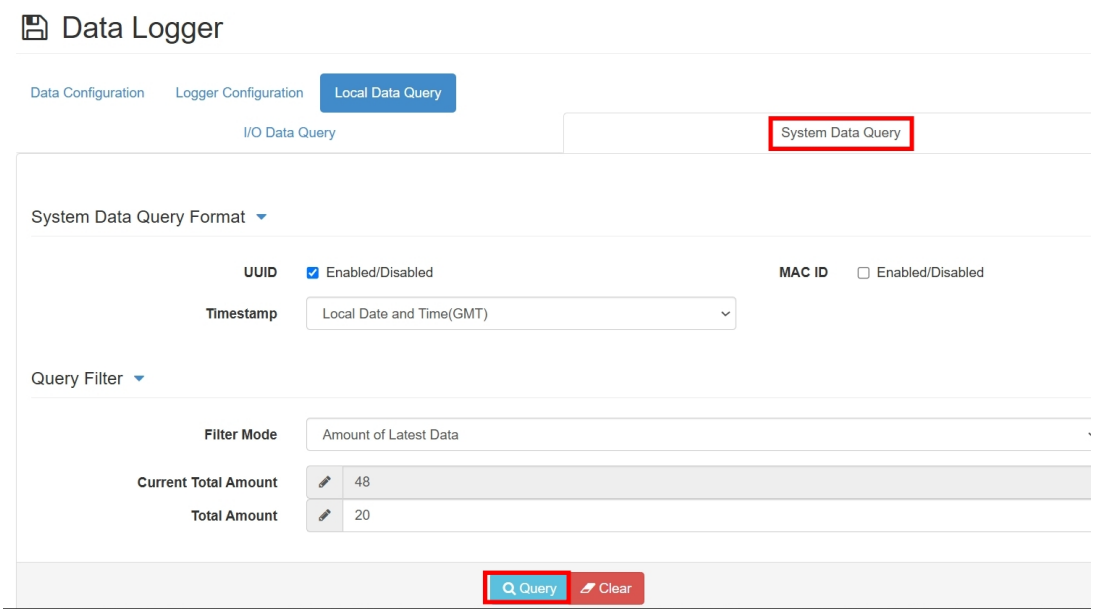
No Filter Enabled: It will query all the data.

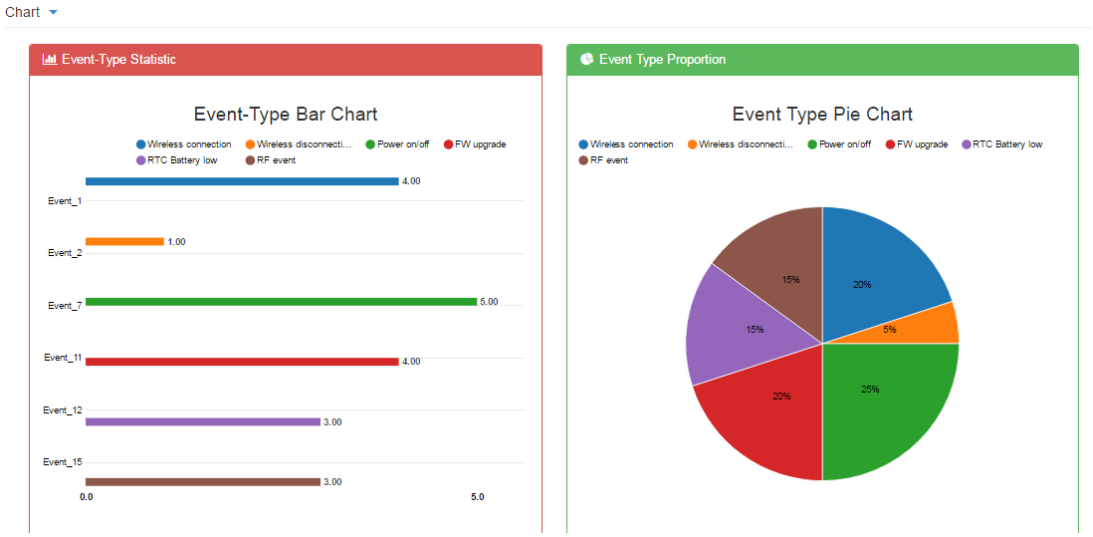
Time Filter: It will query a period of time of data.

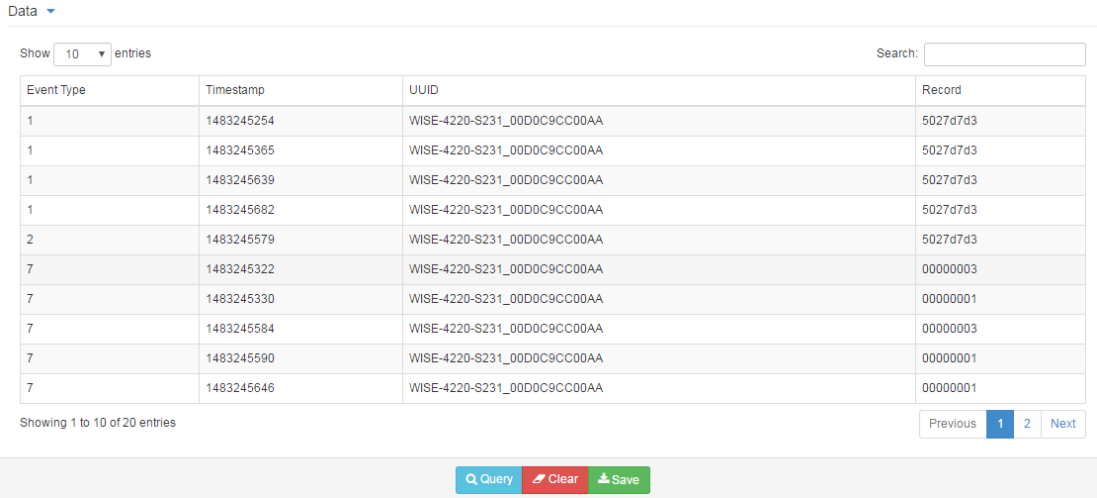
Amount of Latest Data: It will query a quantity of data that you can set by yourself.



Click the Query button to query the data.







4.2.7.4 Cloud Logger

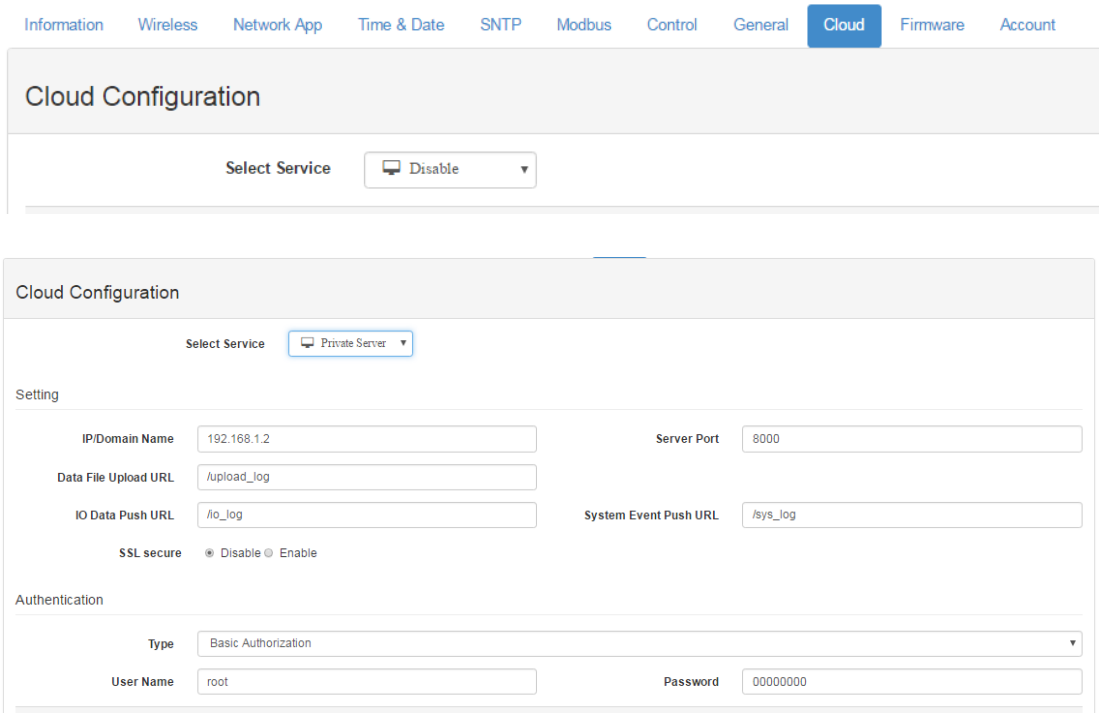
Dropbox

Refer to section **4.2.6 Configuring Cloud Server** for Dropbox cloud logger.

Private Server

If you don't want to push the data to a public file-based cloud like Dropbox, WISE also supports a “Private Server” function which pushes data to a private web server directly. You can setup your own web service to retrieve data from the WISE module, or use the example agent on your own server to retrieve the files pushed from the WISE module.

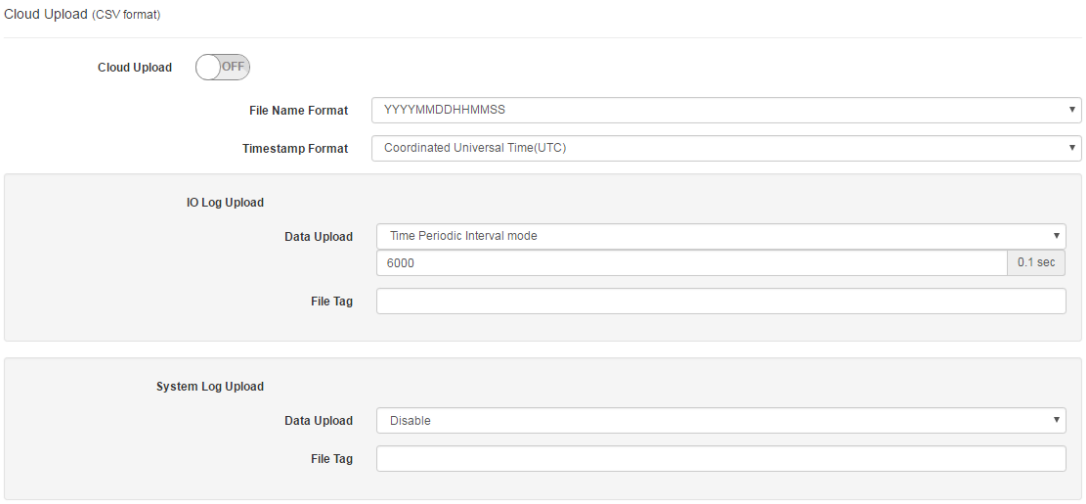
Go to “Cloud Configuration” page and select “Private Server.” Then configure the Private Server Setting. If you would like to use the example agent provided by WISE, you need to confirm the Server IP and Server Port, and make sure the server port you configure is not occupied by another application on your private server. To setup your own application to retrieve files from WISE, you need to configure the URL. We also support SSL security with Dropbox, in order to provide you a safe private cloud solution. Once SSL security is enabled, you need to setup the SSL service on your private server.



If you are not able to setup the SSL service, there is another option for internet security. You can use “Authentication” section for the private server with a User Name and Password provided by our example agent.

After “**Cloud Configuration**” is configured as a private server, you can go back to the **Logger Configuration** page in **Data Logger.** Before switching Cloud Upload to ON, you can configure the data upload criteria for I/O signal and system diagnosis individually.

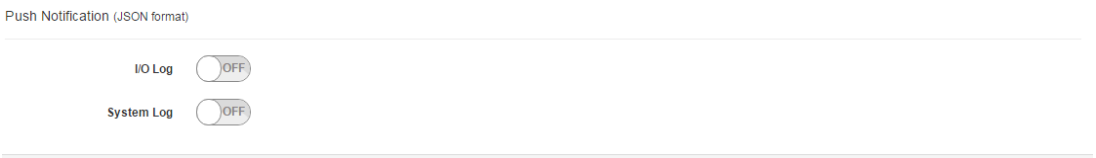
You can choose to push a certain amount of data by using Item Periodic Interval mode, or push data within a certain period of time (Unit: 0.1 sec) by using Time Periodic Interval mode. If you don't want to upload I/O or system data, choose “Disable”. After configuring the upload criteria, you can switch Cloud Upload ON and start uploading. The data will be pushed to the cloud in the format of *.csv.



■ Push Notification

For Cloud Logger functions including Dropbox Cloud Logger and other private servers, data comes from the local memory of WISE. You can pack data from the WISE data logger into a file, and then push it to the web server. Then you can push the latest data when the log condition is triggered with the events such as changes of DI status.

The WISE module will push a notification in JSON format to the private server. You can switch the I/O Log or System Log ON, and then the WISE module will start pushing the latest logged data to the private server.



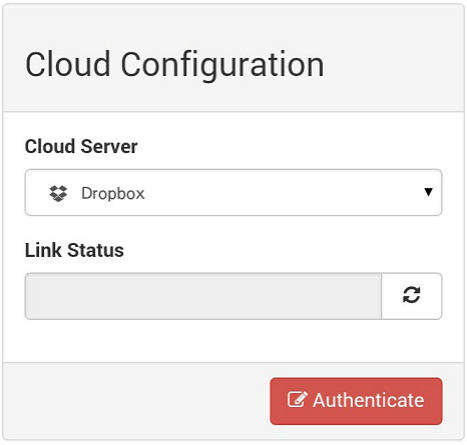
4.2.8 Configuring Cloud Server

1. Make sure the WISE-4250 module is able to access the Internet, and the device that’s going to configure the WISE-4250 module is in the same IP domain as the WISE-4250 module.
2. Go to the Cloud tab of Configuration.

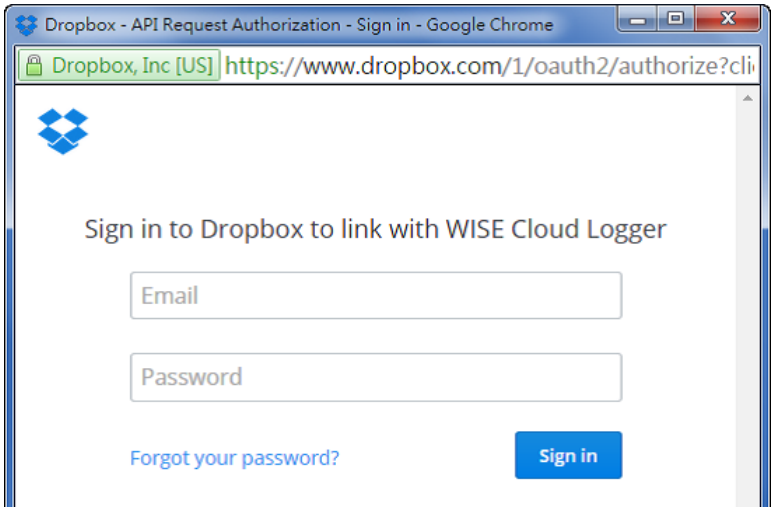
Note! *The following instructions use Dropbox. Make sure Dropbox provide their service in your region or find an alternative public cloud service.*

Note! *Before start to configure cloud sever, please make sure WISE module is working in “Infrastructure Mode”, and the connected AP is able to access internet*

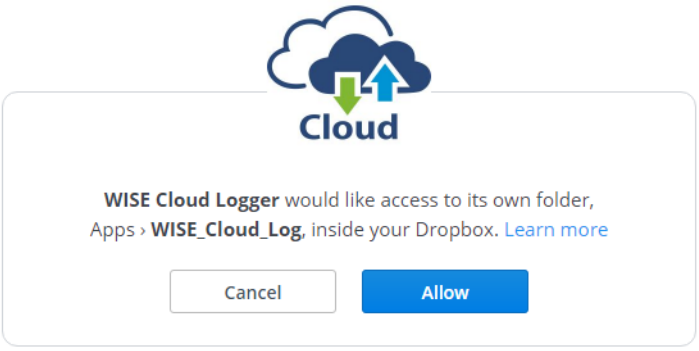
3. Select Dropbox as the cloud server.



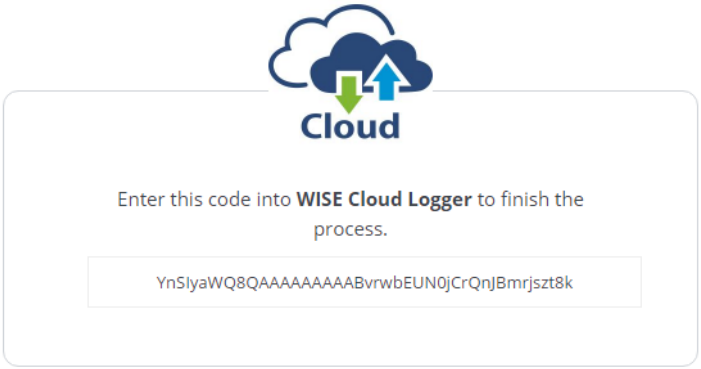
4. The browser will open a new window for Dropbox. Enter your Dropbox account information including E-mail and Password, then click “Sign in.”



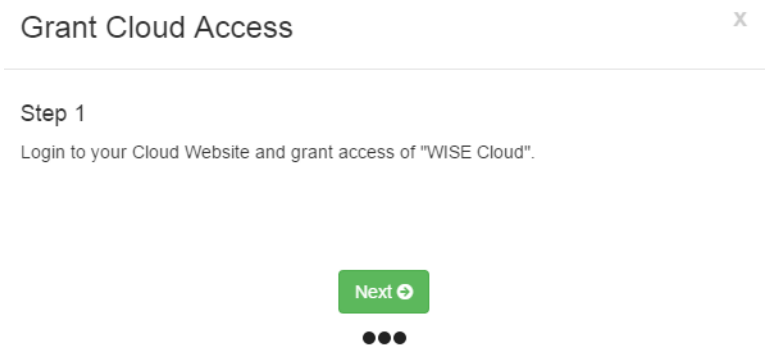
5. After logging in, click "Allow" to allow WISE Cloud Logger Apps to access your Dropbox account to store the data log file.



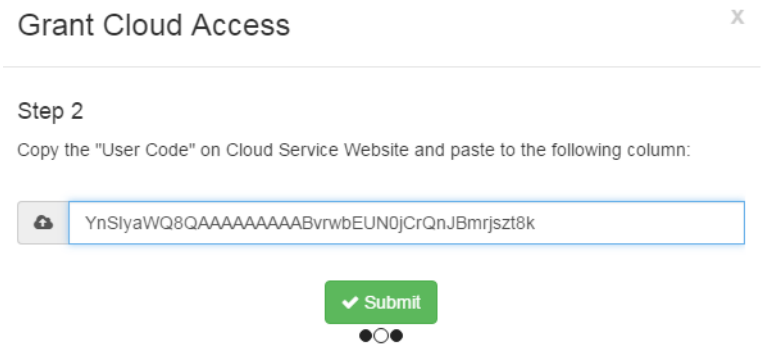
6. Dropbox will then provide a code. Copy this code and return to the configuration web page of the WISE module.



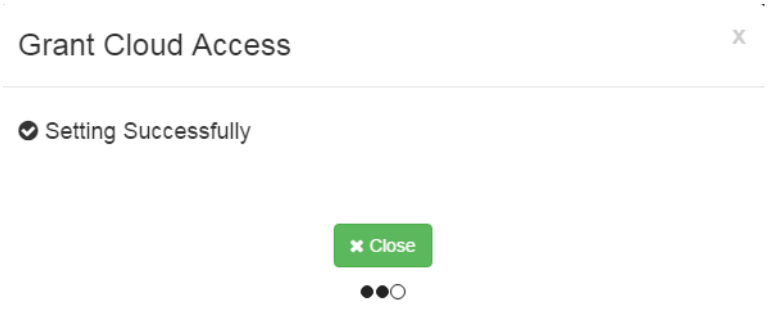
7. Click "Next" to enter the code.



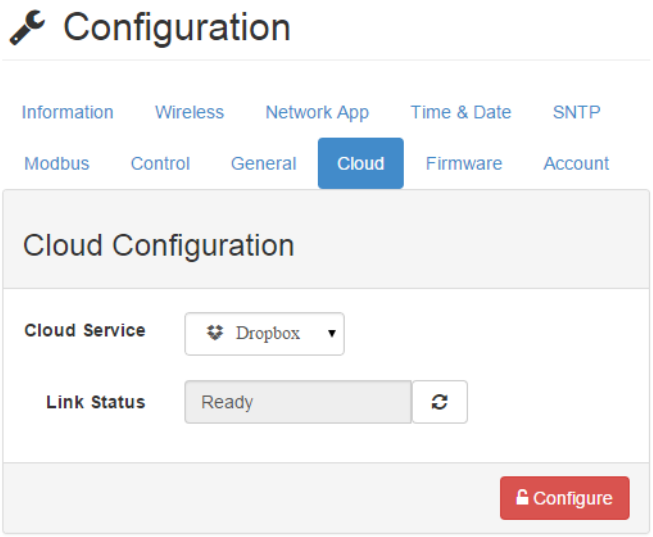
8. Paste the code provided by WISE Cloud Logger, then click "Submit."



9. If your WISE-4250 module is correctly connected to the Internet, you will be able to set the functions successfully. Click "Close" to return to Configuration.

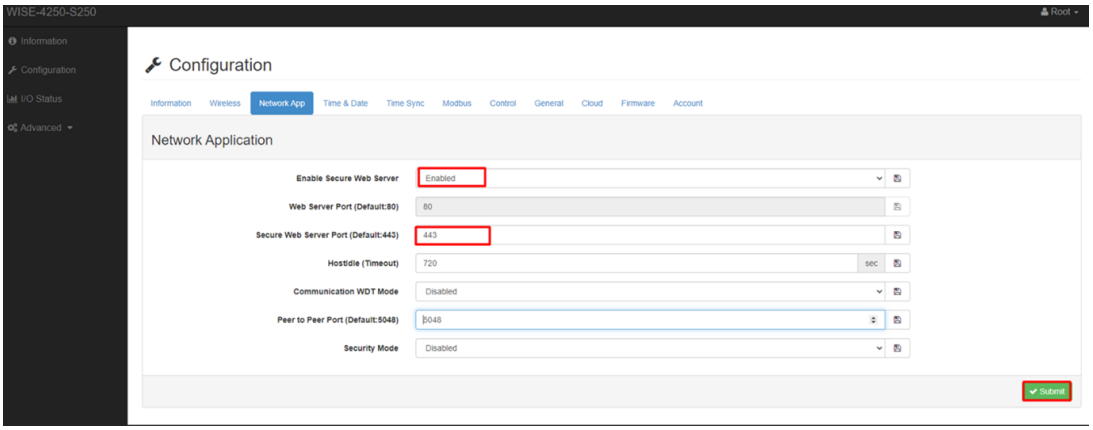


10. You will then see the “Link Status” shows “Ready.”

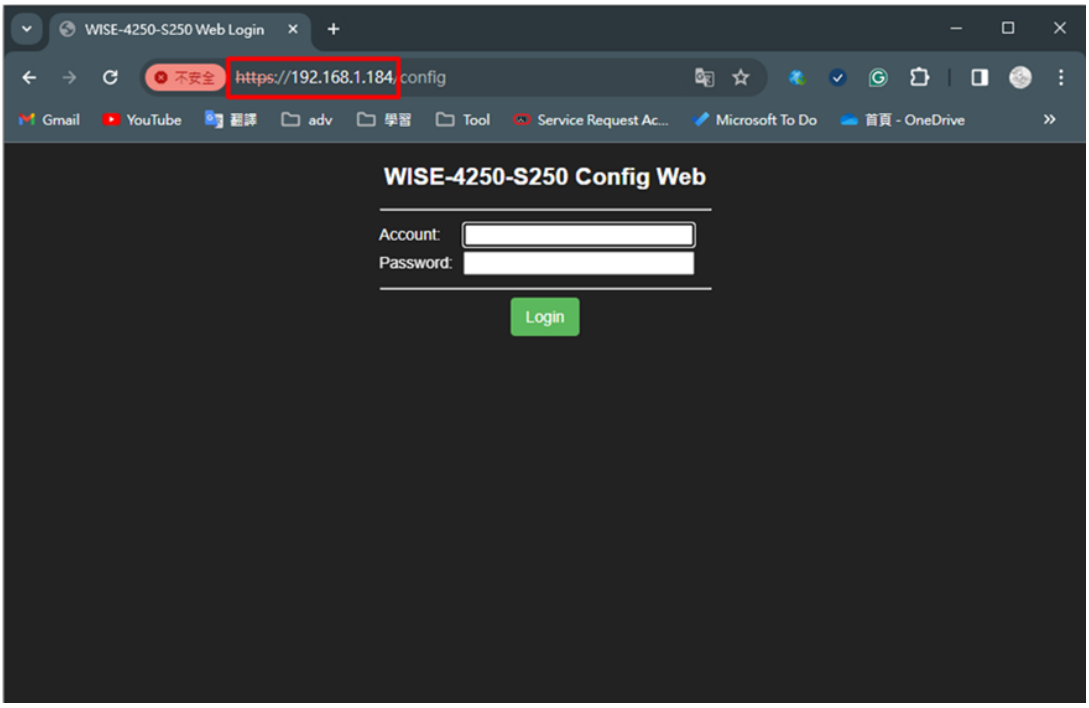


4.2.9 HTTPS

WISE-4250 support Secure Web Server which using port 443.

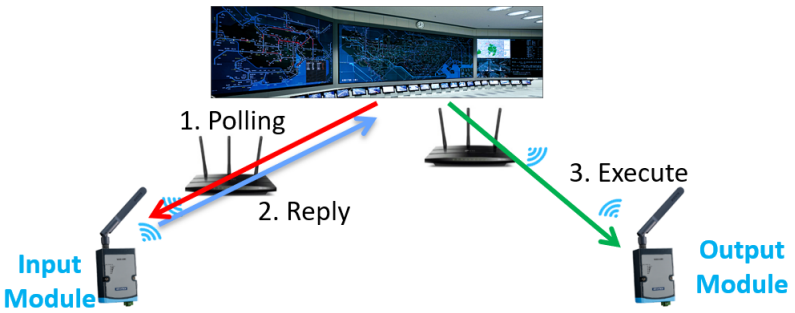


Enter the “https://{WISE-4250 IP} in the browser to enter the WISE-4250 configuration page through secure web server.



4.2.10 Peer to Peer (P2P)

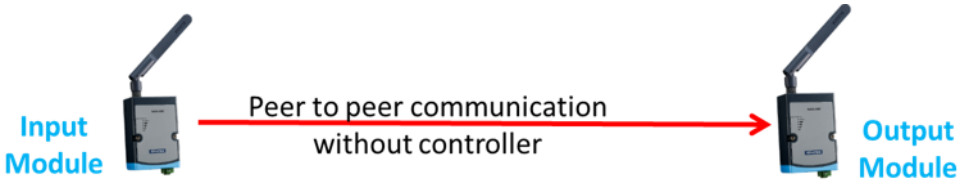
4.2.10.1 Typical Application: Signal Synchronization via Controller



4.2.10.2 What is P2P?

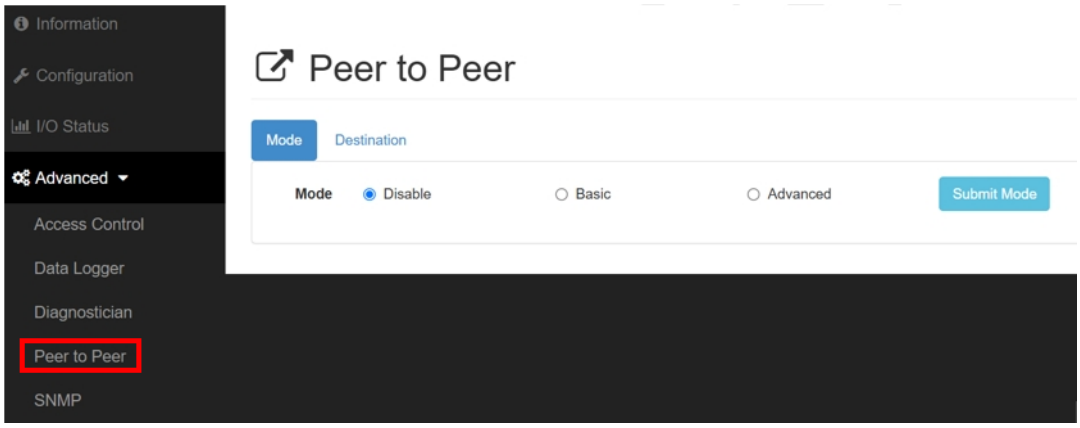
The concept of peer to peer is the input of device A will trigger output of device B by sending control command.

When an input module (Analog Input or Digital Input) sense a signal, it can pass this signal to output module (Analog output or Digital Output) by period or by change of state (C.O.S) which can instantly control your device.



It supports two modes: a basic mode for a single target module/channel and an advanced mode for multiple target modules/channels.

By utilizing P2P technology, modules can communicate directly, effectively reducing latency and improving response time. Furthermore, data transmission uses the UDP protocol (ASCII commands) and can be encrypted with AES-128 to ensure communication security.



■ Basic mode

It is worked with **identical channel number mapping**.

For example: Digital Input channel 1 to Digital Output channel 1 of multiple modules.

Only DI can control DO/relay; AI cannot control DO/relay.

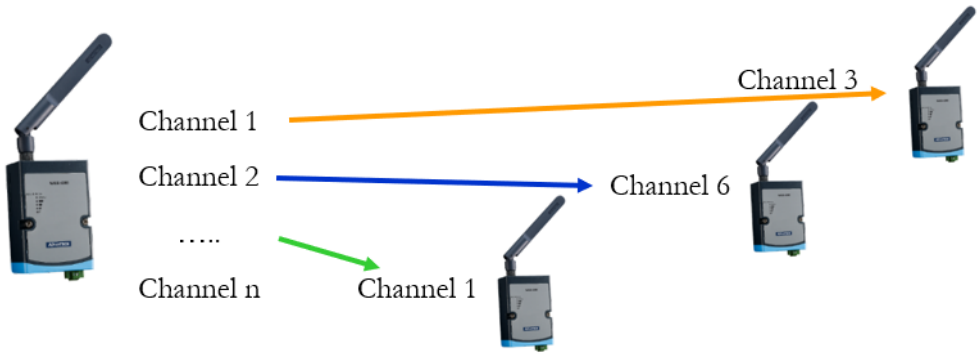
Digital input module map to remote module: In basic mode, user need to be aware of WISE module's Digital Output or relay only can be triggered by Digital Input.

Analog input module can't map to remote module

Advanced mode

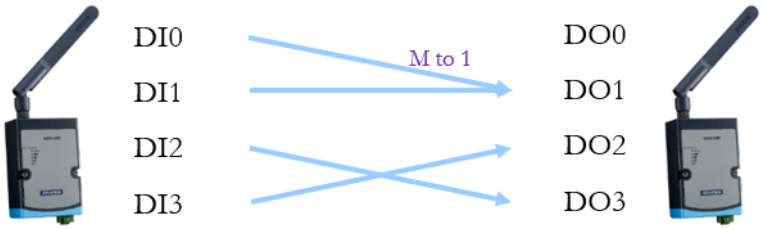
You can use different channel number mapping between different input and output module.

For example: Digital Input channel 1 mapping to Digital Output channel 3 or Digital Input channel 2 mapping to Digital Output channel 6 of another output module as the picture shows.

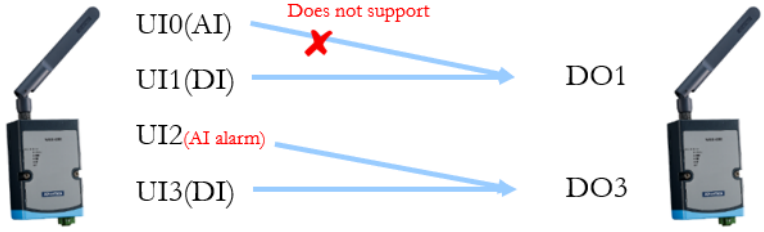


DI can control DO/relay; AI can also control DO/relay only by AI alarm mode.

Digital input module map to DO module



Universal input module map to remote module: The main difference between basic mode and advanced mode is channel number didn't need to be fixed in advanced mode. Because of this difference, user can trigger the Digital Output or relay from multiple inputs.



4.2.10.3 P2P Configuration in Web Utility

■ Basic mode

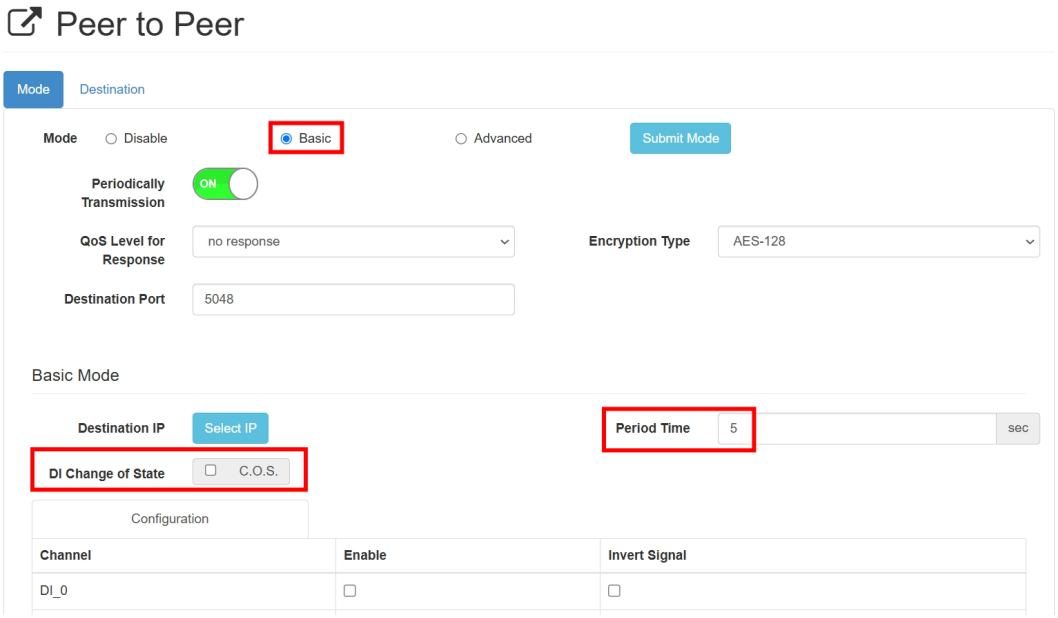
Device can send the data periodically, default of period time is 5 seconds.

User can send data base on Change of State (C.O.S.), which is sending data when DI logic status is changed.

User can setup the customized setting such as QoS level, encryption type, and destination port.

Setting steps:

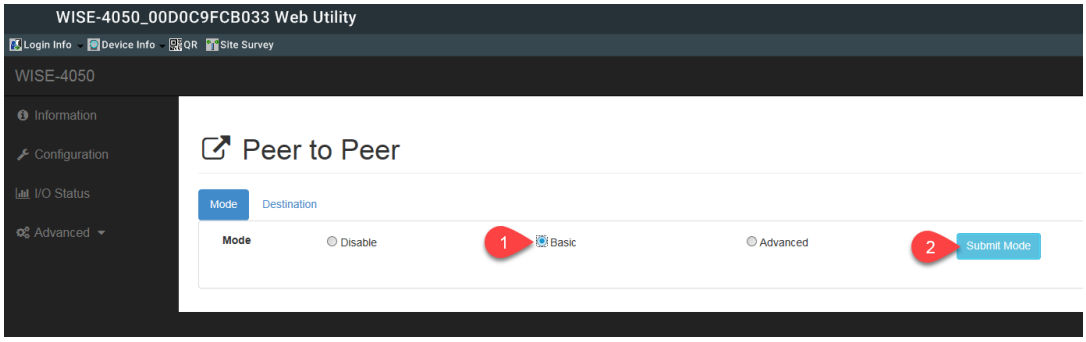
- Choose mode
- Set QOS level
- Set encryption type
- Set destination port
- Setting period
- Assign IP
- Setting individual Channel
- Apply list



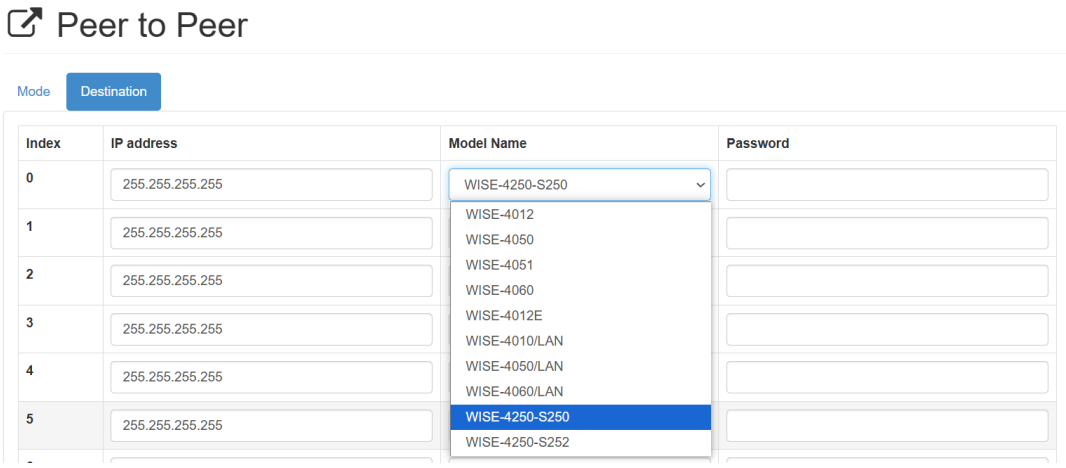
4.2.10.4 P2P Example (Basic Mode)

We use DI of WISE-4250-S251 to trigger DO of WISE-4250-S250. WISE-4250-S251 is set as dry contact, the DI channel 0 status will trigger the status of the WISE-4250-S250 DO channel 0.

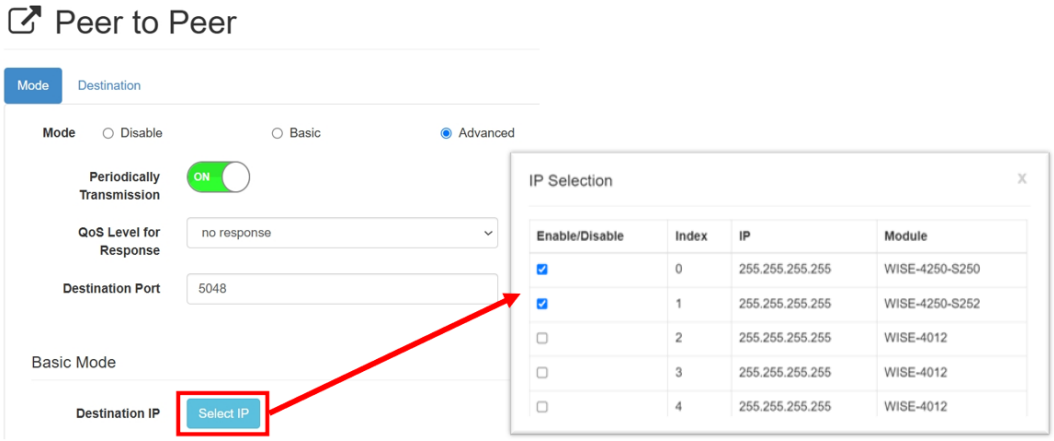
1. Set P2P in Basic mode on input module. (WISE-4250-S251, 192.168.0.100).
2. Click “Submit Mode” to choose basic operating mode.



3. Change to “Destination” tab and type-in the destination module IP, model name and password. (WISE-4250-S250, 192.168.0.102)

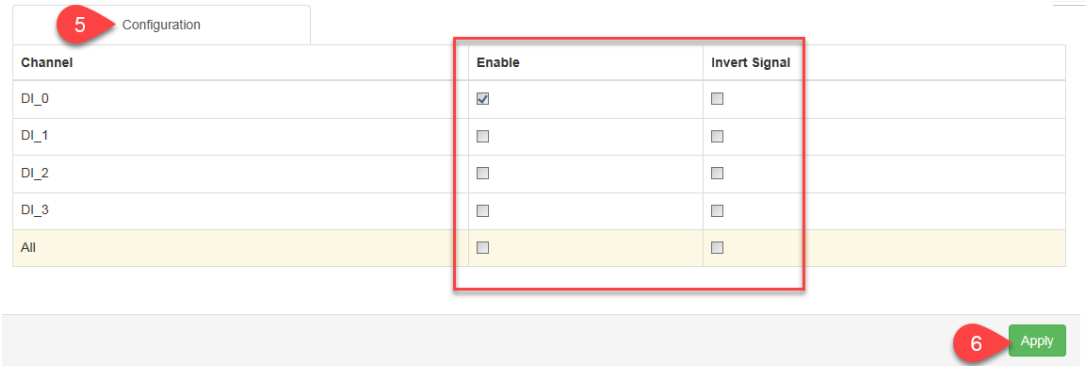


4. Select output module IP address.



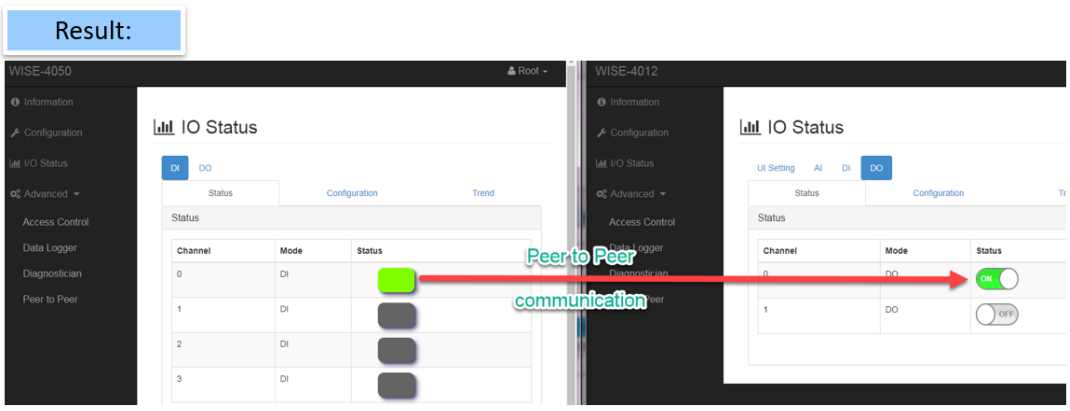
5. Set period time or C.O.S. or user can use both.

6. Select the channel to use P2P function.
7. Click "**Apply**" to finish P2P setting.

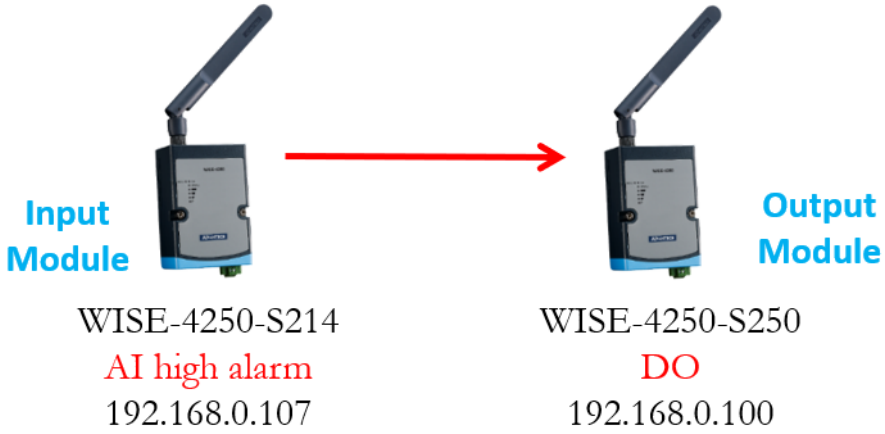


Result:

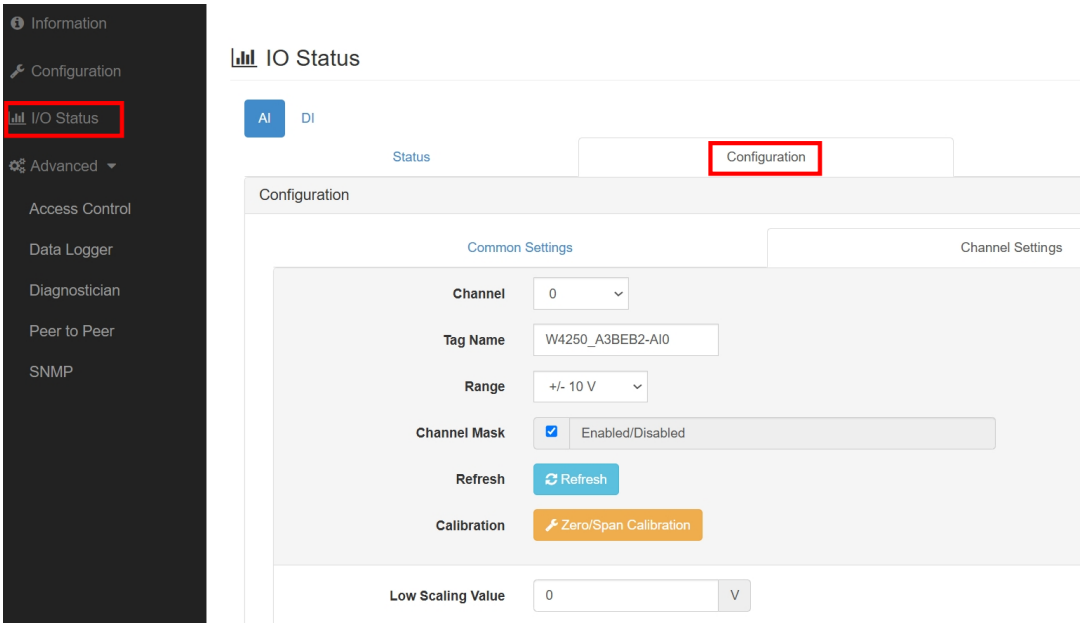
WISE-4250-S251 is set as dry contact, the DI channel 0 status will trigger the status of the WISE-4250-S250 DO channel 0.

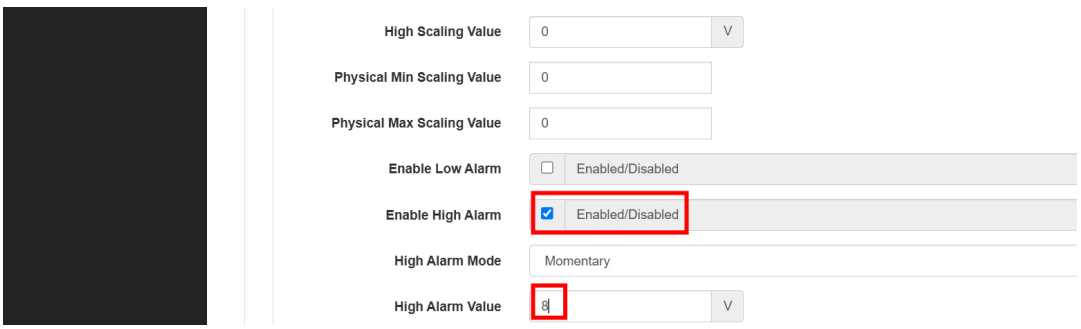


4.2.10.5 P2P Example (Advanced Mode)

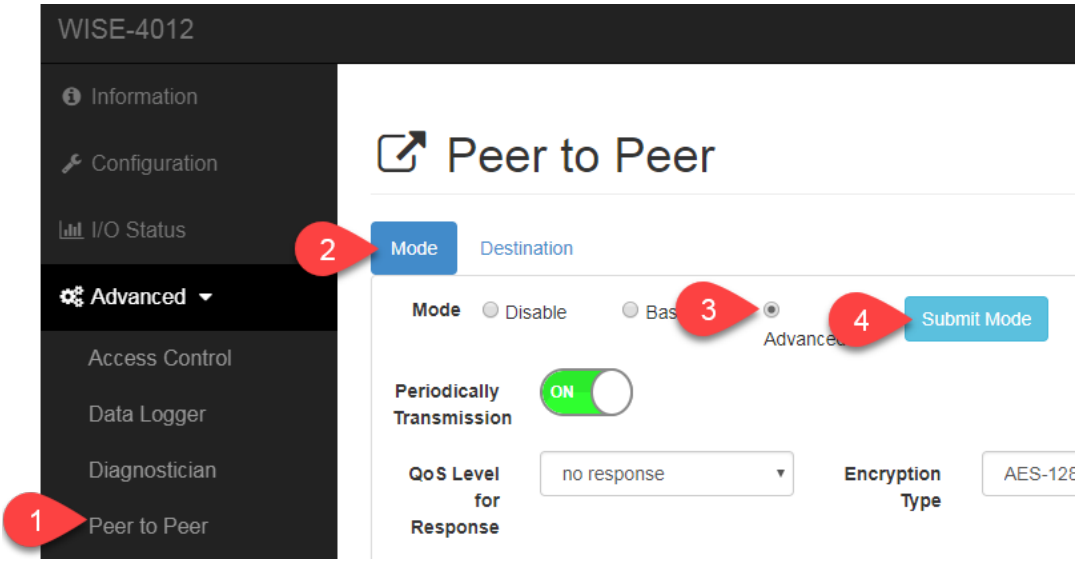


1. Set AI high alarm threshold: We set this AI with high alarm and set the threshold as 8 volts which means when AI value is higher than 8 volts, it will trigger DO of other WISE modules.

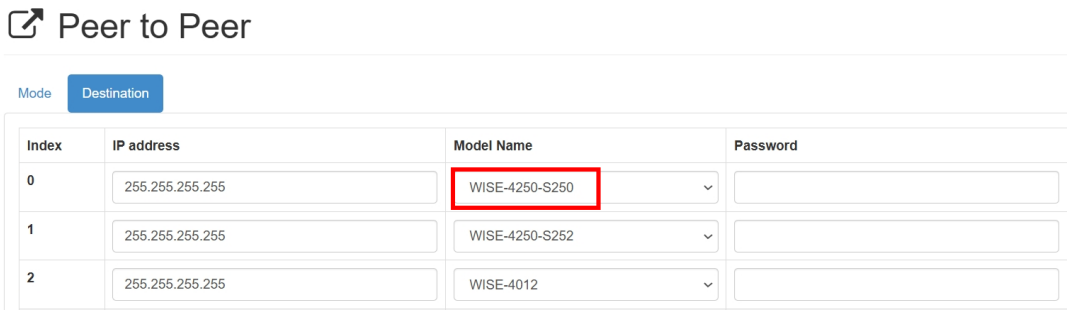




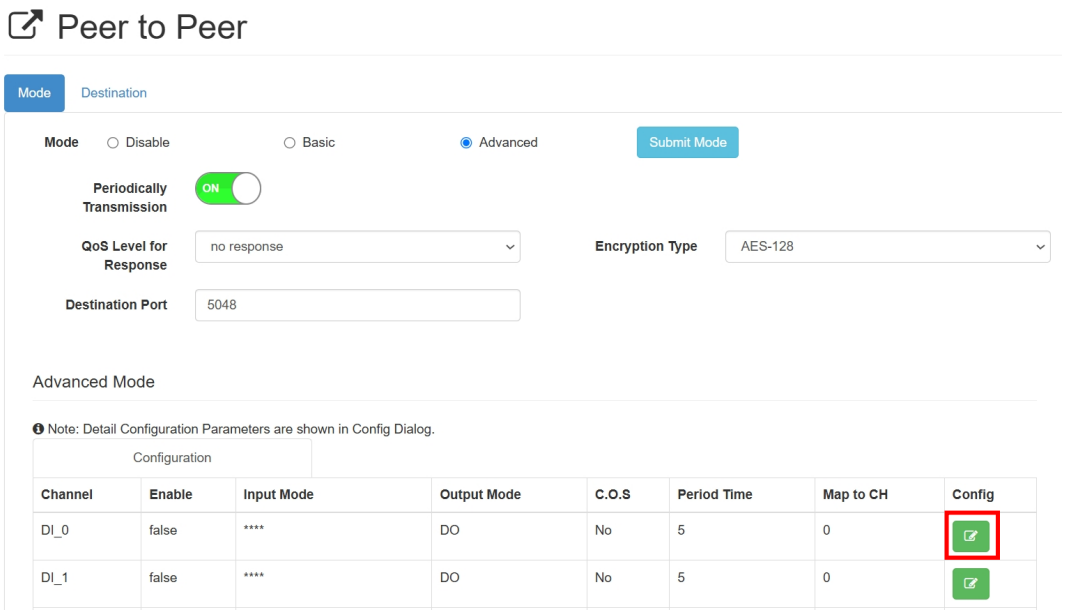
2. Set P2P as Advanced mode on input module. (WISE-4250-S214, 192.168.0.107)
3. Click “Submit Mode” to choose advanced mode.

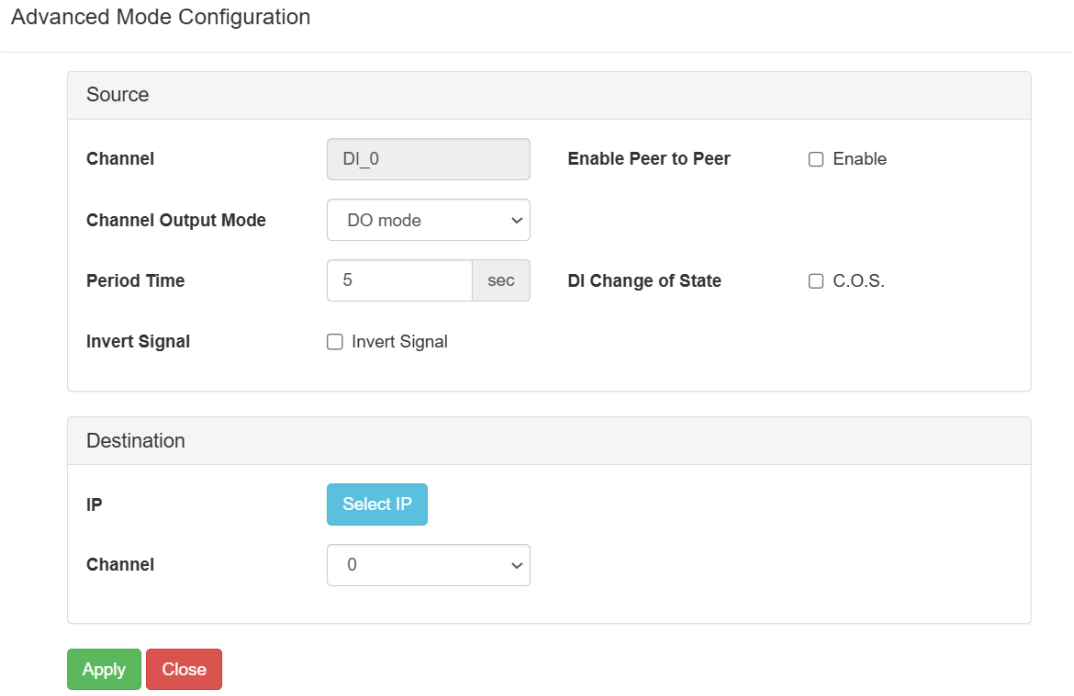


4. Change to “Destination” tab and type-in the destination module IP, model name and password. Then click on “Apply”. (WISE-4250-S250, 192.168.0.100)



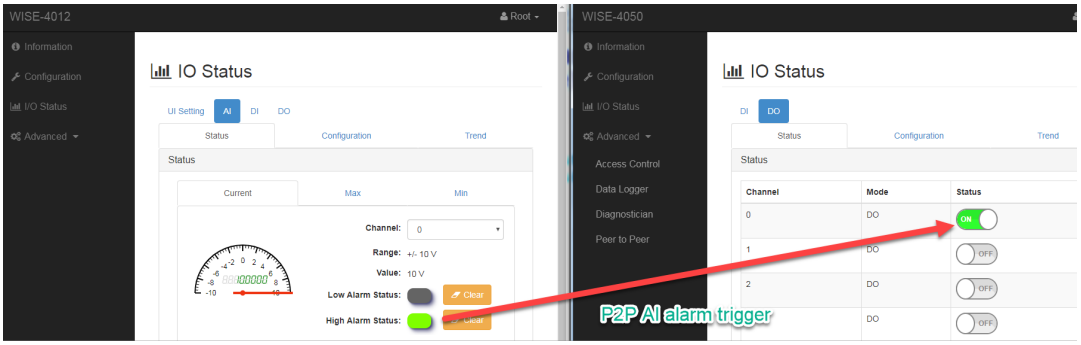
5. Set up the source and the destination, then click on “apply”.
   In this case, AI “high alarm” is used.
   Will trigger DO of WISE-4250-S250 (according to the destination IP).
6. Click “Apply” to finish P2P setting.





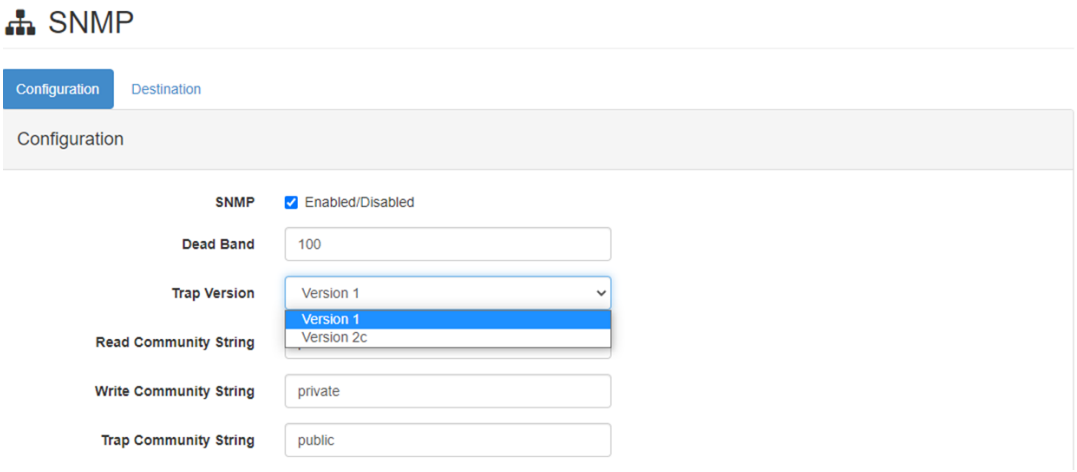
Result:

WISE-4250-S214 AI channel which is alarm mode is going to control the DO of WISE-4250-S250.

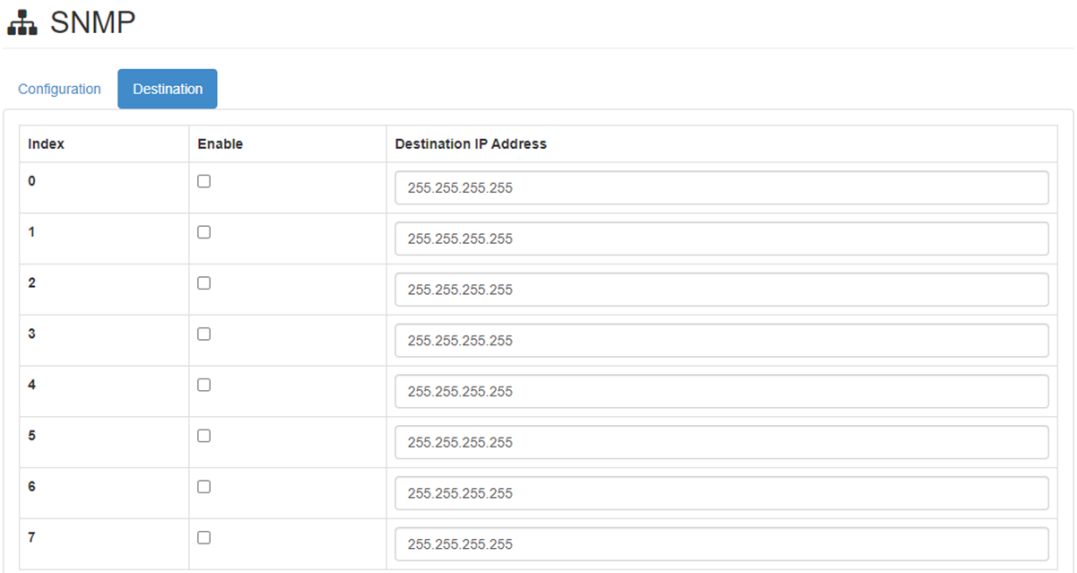


4.2.10.6 SNMP

SNMP function support user to get module information or IO data via SNMP protocol. When IO/Alarm changed, module send trap message to destination IP address set by user. User can parse SNMP message by MIB file.



WISE-4250 can control 16 devices (maximum), if the user turns on or off an I/O on the main device, the other 16 devices would turn or off an I/O.



4.3 General Concept of WISE-4000 Site Survey

How to do the site survey is really important at beginning. In the site survey stage, the **Installation**, **RSSI**, **Communication Quality** are three very basic and important elements. Therefore, following will show how to do the good survey.

4.3.1 Installation

The Wi-Fi signal is influenced by the shelter easily, like pillar, wall and partition. To avoid this kind of barrier, WISE should be put at the higher place where it can be Line-of-Sight (Figure 1) to AP. Line-of-Sight means no obstruction in the middle of point A (antenna 1) and point B (antenna 2). It can perform the better communication interface between AP and WISE. Moreover, WISE can't be wrapped by the metal. The metal around WISE will shield the signal and the signal can't go out.

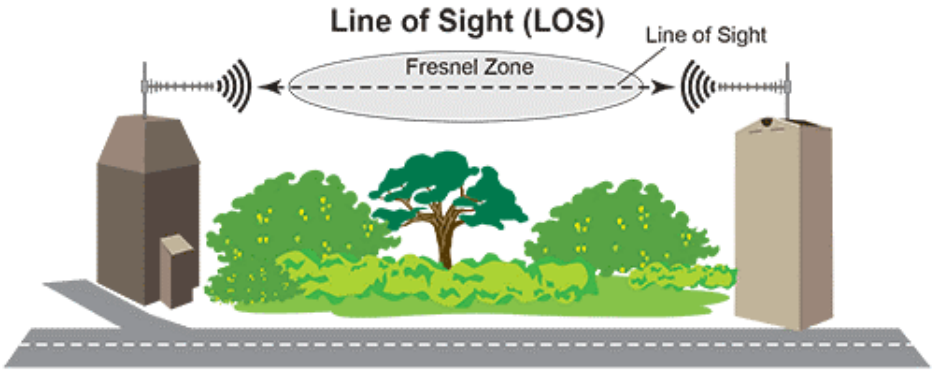


Figure 4.1 Line of Sight (LOS)

4.3.2 RSSI

The relationship between the Wi-Fi signal strength and the number of LED(s) on the panel of WISE-4000 module shows in Figure 2. To achieve good communication quality of Wi-Fi environment, suggest to keep the RSSI above -64dBm, which means the WISE-4000 module should at least shows 3 LEDs. Because the AP plays an important role in Wi-Fi environment, all wireless clients will leverage the AP ability to connect to the outer network.

#LED	RSSI Range
4	RSSI > -48dBm
3	-48dBm > RSSI > -64dBm
2	-64dBm > RSSI > -72dBm
1	-72dBm > RSSI

Figure 4.2 The Relationship between RSSI Level and LED Indicators on WISE-4000 Module

Therefore, when there's a RSSI issue, it is better to determine whether **ALL** or **Just a Few** are facing the poor signal issue. That means:

- If **ALL** of the RSSI of Wi-Fi clients (such as WISE) are poor, it probably means the position of AP is not correct installed and the signal of AP can't reach to

wireless client. Therefore, in that case, it is better to adjust the position/attitude/angle of AP to make the signal cover all of the wireless client.

- If **Just a Few** RSSI of Wi-Fi clients are bad, there might be have several reasons. Here provides step by step to help user to narrow down this problem.

1. Adjust the Tx Power of AP

Tx power influences the signal coverage. If Tx power is stronger, then the signal coverage can be wider. The industrial AP usually has the function to adjust the Tx power to fit the site environment. Figure 2 shows the Tx power setting section of AP EKI-6332GN-AE as example.

Figure 4.3 Tx Power Setting of EKI-6332GN-AE AP

2. Adjust WISE Installation
   Please try to adjust different position/attitude/antenna angle of WISE. Because antenna of wireless devices (AP) has omni-directional and directional type. Some installation of wireless client (WISE) doesn't fit to the type of antenna. If the type of antenna of AP is omni-directional. The attitude/angle of WISE are the key point to influence the RSSI. If the type of antenna of AP is directional. The position/angle is the key point to influence the RSSI.
3. Adjust the Distance between WISE and AP
   Move WISE close to AP and observe whether the signal strength of WISE's LED improved or NOT. If it does improve, the distance (i.e. or interference where WISE being installed) are the possible reasons to cause this. It means that the signal of AP/WISE can't reach to each other.
4. Enhance the Signal Strength
   Based on the step (3), if WISE cannot be moved due to environmental limitation, then change antenna of AP or add extra AP is another way to enhance the signal strength between the AP and wireless client device (WISE).

4.3.3 Communication Quality

After checking the Installation and RSSI value, there is a site survey tool to verify the communication quality between WISE and AP. This tool can keep sending packet to the WISE and receive ACK to diagnosis whether losing data. As well as Average Response Time, Good Packets Percentage, Average Signal Strength(dBm) etc. Relevant details will introduce in Section 2.

4.4 How to Execute the Site Survey Tool of WISE in WISE Studio?

Site Survey is very important to make sure that the stability connection between AP and WISE. WISE has a site survey tool to let customer to do this test. Because this tool is to test the connection stability between AP and WISE, the connection between AP and the test equipment need to be stable. Therefore, the PC with WISE Studio software and the AP must connect to each other by **LAN port**. The topology shows in Figure 4.

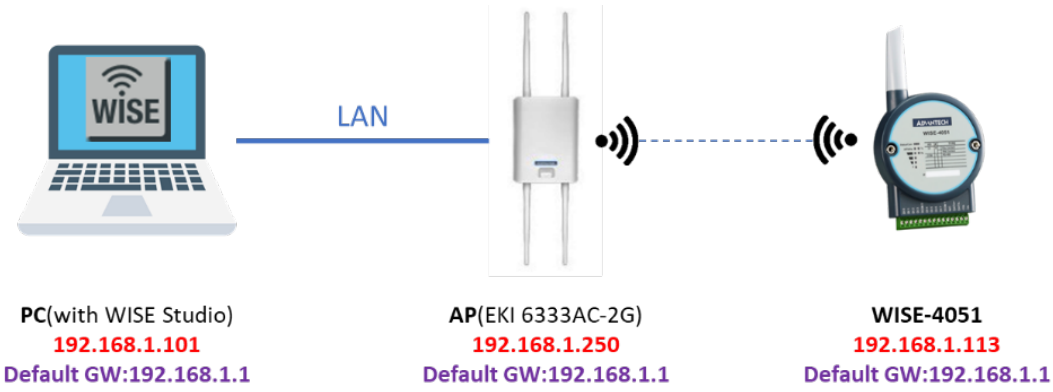


Figure 4.4 Topology of WISE Site Survey

Next, follow the steps below to execute the Site Survey Tool of WISE in WISE Studio.

1. Open the WISE Studio software from personal computer.

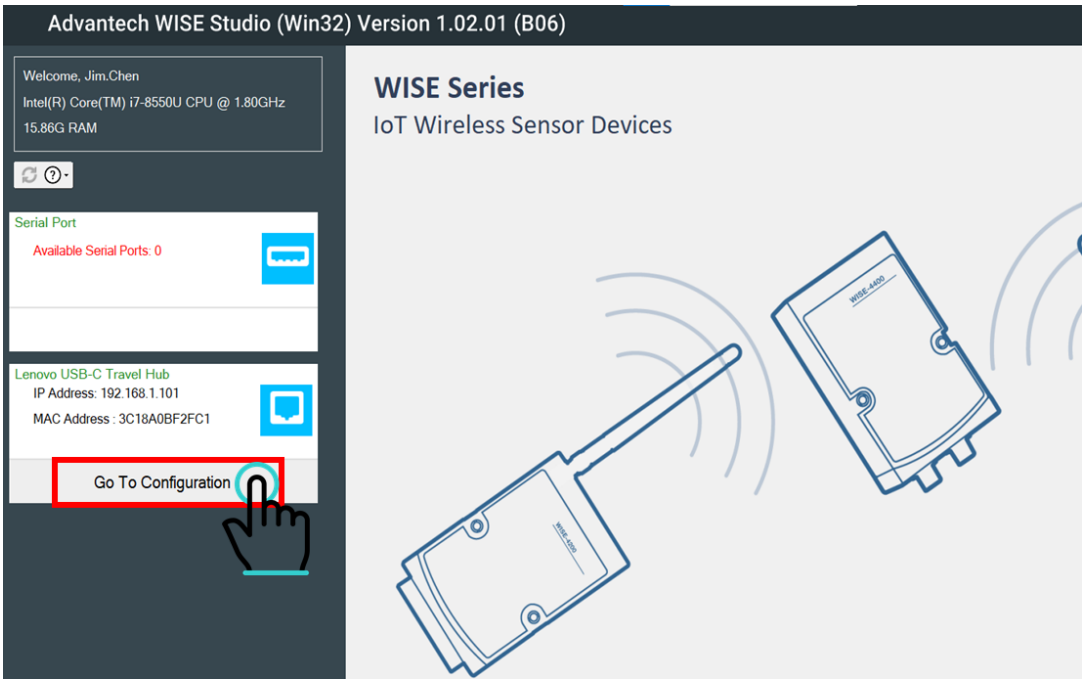


Figure 4.5 WISE Studio Entry Page

Download WISE Studio Software
https://www.advantech.com/zh-tw/support/details/utility?id=1-1MJSJKX

2. Find the Device from WISE Studio Software

a. Click “Go To Configuration” button in Figure 5.

b. Wait for software scanning the device. The device will display as Figure 6.

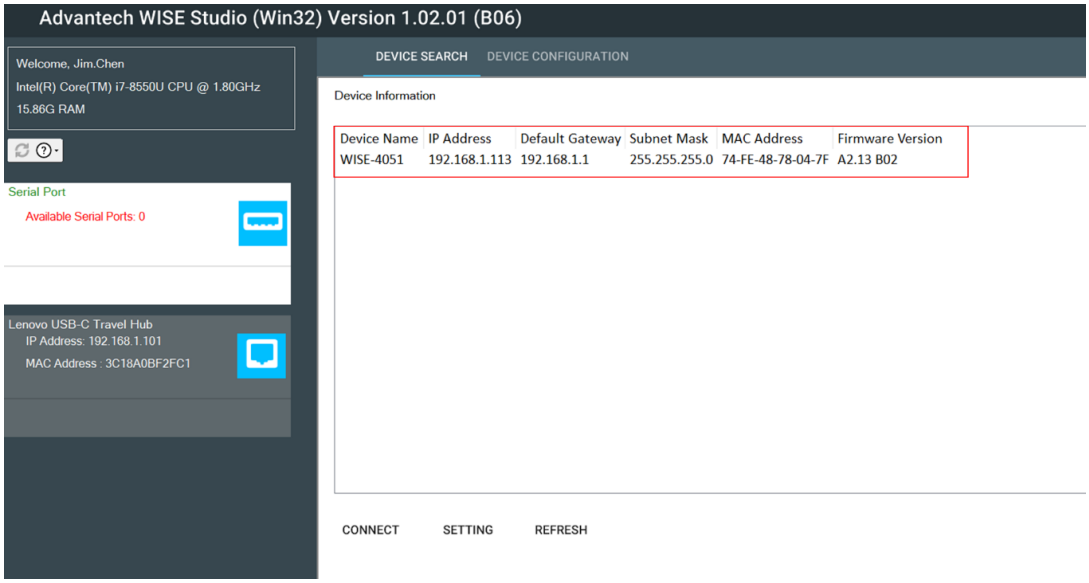


Figure 4.6 Scan the device

3. Use WISE Studio Connect the WISE Device.

a. Select your WISE device and Click “Go To CONNECT” button in Figure 7.

b. The entry page of WISE device will come out as Figure 8.

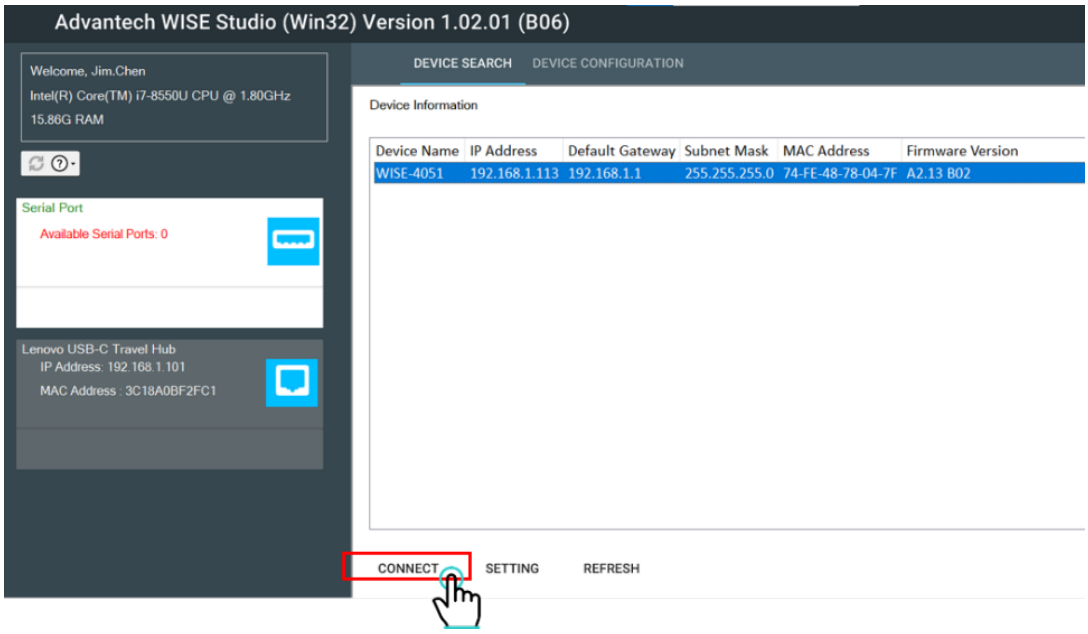


Figure 4.7 Connect to the WISE Device

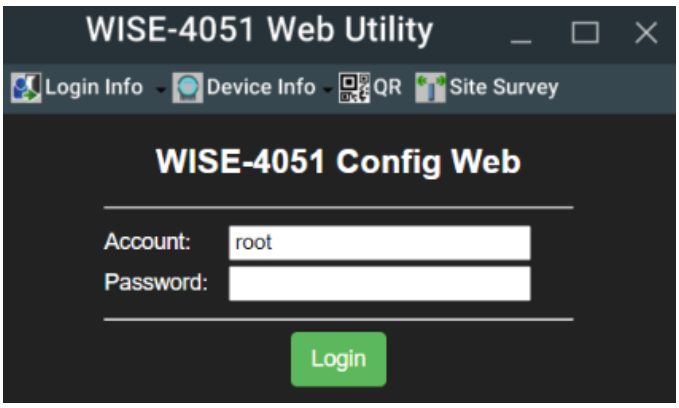


Figure 4.8 Entry Page of WISE Device

4. Execute the Site Survey Tool.

a. Click "**Site Survey**" button in Entry Page of WISE device, shows in Figure 9.

b. Enter to Site Survey Page and click "**START**" button to start the site survey, shows in Figure 10.

c. When start to execute site survey, the LED indicators on WISE-4000 will start to blink and user will not be allowed to enter to the WISE-4000 module.

Figure 4.9 Site Survey Button

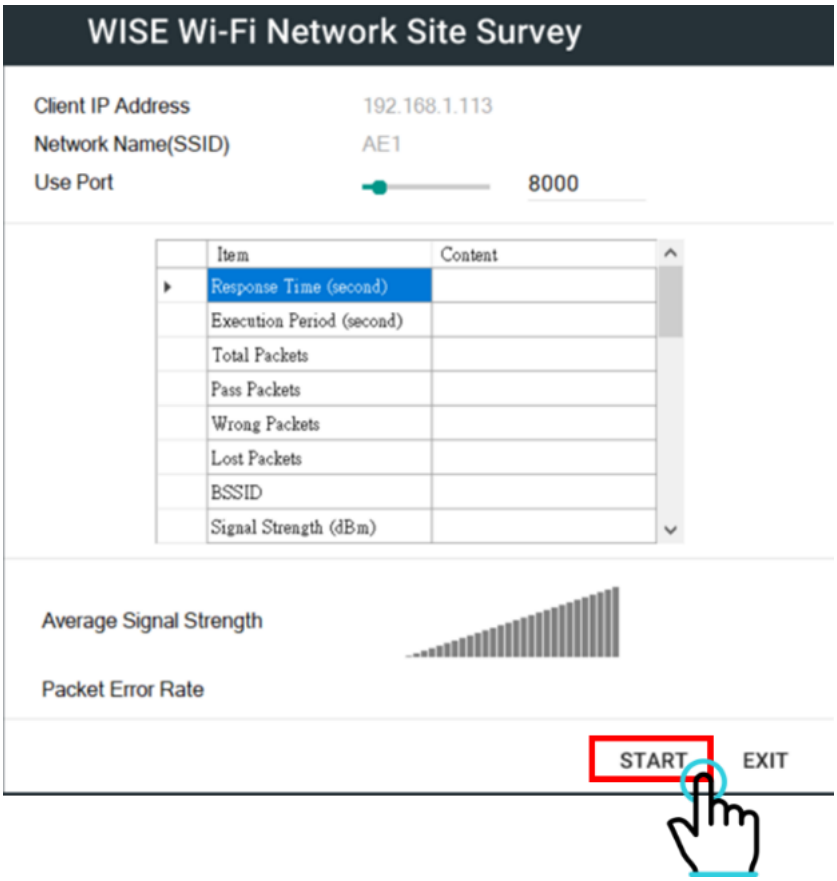


Figure 4.10 Site Survey Page

5. Troubleshooting

If there is no data shown in WISE studio site survey page, user need to enable "Advantech Public" firewall setting of laptop. User can enter this page by this path: **Control Panel\System and Security\Windows Defender Firewall\Allowed apps.**

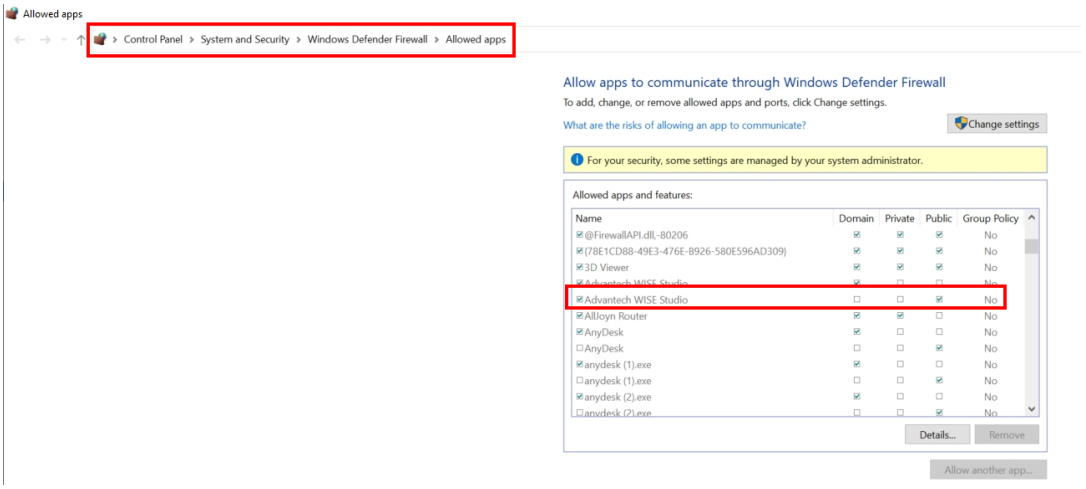


Figure 4.11 Enable Firewall Setting

6. Result of Site Survey

After clicking “START” button to start sending test packets and test for 10 mins to get more reliable average data. The result will show as Figure 11. Please noticed that the Average Signal Strength (RSSI) and Packet Error Rate these two indicators. RSSI value can refer to Figure 2 and make this indicator number above -64dBm as possible as you can. The Packet Lost Rate indicates the rate of lost packets over total packets. The smaller the Packet Lost Rate is, the better transmission status is.

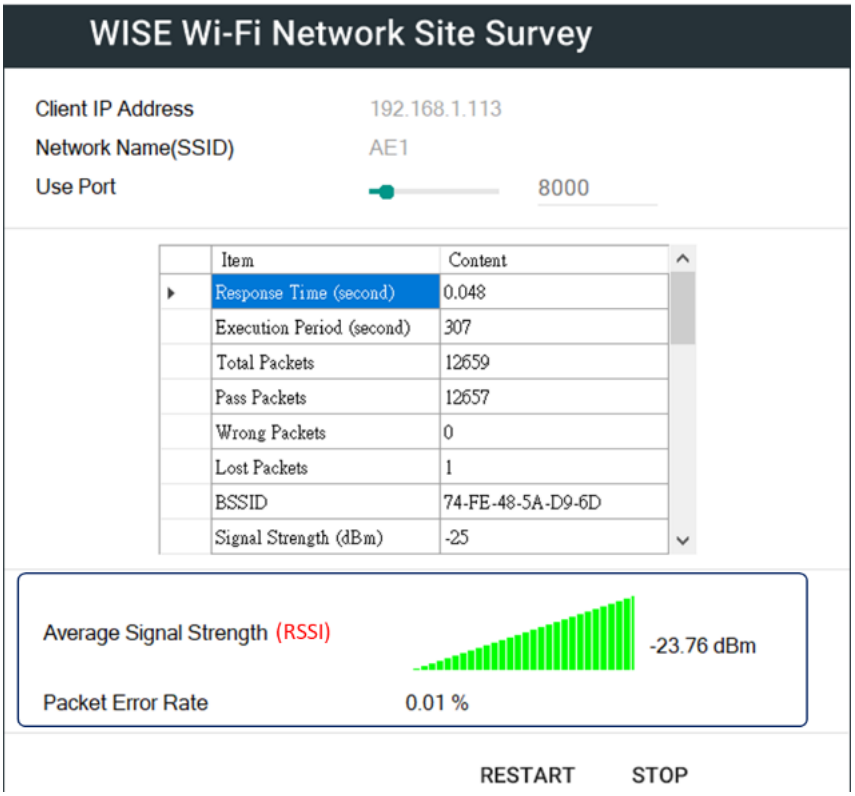


Figure 4.12 Result of Site Survey

Appendix A

Modbus Mapping Table

A.1 Modbus Function Code Introduction

Table A.1: System Configuration File		
Code (Hex)	**Name**	**Usage**
01	Read Coil Status	Read Discrete Output Bit
02	Read Input Status	Read Discrete Input Bit
03	Read Holding Registers	Read 16-bit register. Used to read integer or floating point process data.
04	Read Input Registers	
05	Force Single Coil	Write data to force coil ON/OFF
06	Preset Single Register	Write data in 16-bit integer format
08	Loopback Diagnosis	Diagnostic testing of the communication port
0F	Force Multiple Coils	Write multiple data to force coil ON/OFF
10	Preset Multiple Registers	Write multiple data in 16-bit integer format

A.2 Modbus Mapping Table

A.2.1 WISE-4250-S232 (Coming soon)

Table A.2: WISE-4250-S231 Modbus Mapping Table							
Address 0X	**Ch**	**Description**	**Attribute**	**Address 4X**	**Ch**	**Description**	**Attribute**
00101	0	Reset Historical Max. Sensor Value	Write	**40001**	0	Sensor Value (*2)	Read
00102	1		Write	**40002**	0		Read
00111	0	Reset Historical Min. Sensor Value	Write	**40021**	0	Historical Max. Sensor Value (*2)	Read
00112	1		Write	**40022**	0		Read
00131	0	High Alarm Flag	Read	**40041**	0	Historical Min. Sensor Value (*2)	Read
00132	1		Read	**40042**	0		Read
00141	0	Low Alarm Flag	Read	**40101 ~40102**	0	Sensor Status (*1)	Read
00142	1		Read	**40103 ~40104**	1		Read
				40201	0	**Type Code	Read
				40202	1		Read
Address 0X	**Ch**	**Description**	**Attribute**	**Address 4X**	**Ch**	**Description**	**Attribute**
05001		RTC Battery Low	Read	45101		Data Logger Status	Read
				45302		RSSI	Read
Address 0X	**Ch**	**Description**	**Attribute**	**Address 4X**	**Ch**	**Description**	**Attribute**
				40211		Module Name 1	Read
				40212		Module Name 2	Read
				40213		Module Name 3	Read
				40214		Reserved for Module Name	Read

Table A.3: **Data format defined by sensor		
Sensor code	**Sensor value**	**Data**
0x1000	Temperature (ºC)	signed short
0x1001	Temperature (ºF)	signed short
0x1002	Temperature (K)	signed short
0x1020	Humidity	signed short

Table A.4: *1 AI/Sensor Status: (1st Register at Low Address)		
Bit Order	**Description**	**I/O Type**
0	Fail to provide AI value or sensor data	AI, AO Sensor

Table A.5: *2 Sensor Type Code	
Type Code	**Input range**
0x1000	Temperature (ºC)
0x1001	Temperature (°F)
0x1002	Temperature (K)
0x1020	Humidity (%)

A.2.2 WISE-4250-S214

Table A.6: WISE-4250-S214							
Sensor (Max 10)		**0**	**DI (Max 12)**		**4**	**DO (Max 12)**	**0**
AI (Max 12)		**4**	**COM(MAX 2)**		**0**		
DI							
Address 0X	**Ch**	**Description**	**Attribute**	**Address 4X**	**Ch**	**Description**	**Attribute**
00001	0	DI Value	Read	**40017~40018**	0	Counter/Frequency Value	Read
00002	1		Read	**40019~40020**	1		Read
00003	2		Read	**40021~40022**	2		Read
00004	3		Read	**40023~40024**	3		Read
00033	0	Counter Start(1)/ Stop(0)	R/W	**40301**	All	DI Value	Read
00034	1		R/W				
00035	2		R/W				
00036	3		R/W				
00037	0	Clear Counter(1)	Write				
00038	1		Write				
00039	2		Write				
00040	3		Write				
00041	0	Clear Overflow	R/W				
00042	1		R/W				
00043	2		R/W				
00044	3		R/W				
00045	0	DI Low Latch Status	R/W				
00046	1		R/W				
00047	2		R/W				
00048	3		R/W				

Table A.6: WISE-4250-S214							
00049	0	DI High Latch Status	R/W				
00050	1		R/W				
00051	2		R/W				
00052	3		R/W				
AI							
Address 0X	**Ch**	**Description**	**Attribute**	**Address 4X**	**Ch**	**Description**	**Attribute**
00101	0	Reset Historical Max. AI Value	Write	**40001**	0	AI Value	Read
00102	1		Write	**40002**	1		Read
00103	2		Write	**40003**	2		Read
00104	3		Write	**40004**	3		Read
00105	Average Ch 0~3		Write	**40005**	Average Ch 0~3		Read
00111	0	Reset Historical Min. AI Value	Write	**40101~40102**	0	AI Status (*1)	Read
00112	1		Write	**40103~40104**	1		Read
00113	2		Write	**40105~40106**	2		Read
00114	3		Write	**40107~40108**	3		Read
00115	Average Ch 0~3		Write				
00121	0	Open-Circuit Flag (Burnout)	Read	**40111**	0	Historical Max. AI Value	Read
00122	1		Read	**40112**	1		Read
00123	2		Read	**40113**	2		Read
00124	3		Read	**40114**	3		Read
				40115	Average Ch 0~3		Read
00131	0	High Alarm Flag	Read	**40121**	0	Historical Min. AI Value	Read
00132	1		Read	**40122**	1		Read
00133	2		Read	**40123**	2		Read
00134	3		Read	**40124**	3		Read
00135	Average Ch 0~3		Read	**40125**	Average Ch 0~3		Read
00141	0	Low Alarm Flag	Read	**40131~40132**	0	AI Floating Value (IEEE754)	Read
00142	1		Read	**40133~40134**	1		Read
00143	2		Read	**40135~40136**	2		Read
00144	3		Read	**40137~40138**	3		Read
00145	Average Ch 0~3		Read	**40139~40140**	Average Ch 0~3		Read
				40151~40152	0	Historical Max. AI Floating Value (IEEE754)	Read
				40153~40154	1		Read
				40155~40156	2		Read
				40157~40158	3		Read
				40159~40160	Average Ch 0~3		Read
				40171~40172	0	Historical Min. AI Floating Value (IEEE754)	Read
				40173~40174	1		Read
				40175~40176	2		Read
				40177~40178	3		Read
				40179~40180	Average Ch 0~3		Read

Table A.6: WISE-4250-S214							
				40191	0	AI value after scaling	Read
				40192	1		Read
				40193	2		Read
				40194	3		Read
				40195	Average Ch 0~3		Read
				40201	0	Type Code (*3) (The type codes of channels for average value can't be changed.)	R/W
				40202	1		R/W
				40203	2		R/W
				40204	3		R/W
				40205	Average Ch 0~3		Read
				40221	All	AI Channel Enable	R/W
				40231~40232	0	Physical AI Floating Value (IEEE754)	Read
				40233~40234	1		Read
				40235~40236	2		Read
				40237~40238	3		Read
				40239~40240	Average Ch 0~3		Read
Miscellaneous							
Address 0X	**Ch**	**Description**	**Attribute**	**Address 4X**	**Ch**	**Description**	**Attribute**
5001		RTC Battery Low	Read	**45101**		Data Logger Status	Read
				45302		RSSI	Read
Addresses for Internal Use							
Address 0X	**Ch**	**Description**	**Attribute**	**Address 4X**	**Ch**	**Description**	**Attribute**
				40211		Module Name 1	Read
				40212		Module Name 2	Read
				40213		Module Name 3	Read
				40214		Reserved for Module Name	Read

Table A.7: *1 AI/Sensor Status: (1st Register at Low Address)		
Bit Order	**Description**	**I/O Type**
0	Fail to provide AI value or sensor data	AI, AO Sensor
1	Over Range	AI
2	Under Range	AI
3	Open Circuit (Burnout)	AI
4	Reserved	
5	Unavailable Channel Configuration	AI
6	Reserved	
7	ADC initializing/Error	AI
8	Reserved	
9	Zero/Span Calibration Error	AI, AO
10~15	Reserved	

Table A.8: *1 AI Status: (2nd Register at High Address)		
Bit Order	**Description**	**I/O Type**
0	DI triggered to Safety Value	AO
1	DI triggered to Startup Value	AO
2~15	Reserved	AO

Table A.9: *3 AI Type Code	
Type Code	**Input range**
0x0148	0~10V
0x0147	0~5V
0x0145	0~1V
0x0106	0~500mV
0x0105	0~150mV
0x0143	+/-10V
0x0142	+/-5V
0x0140	+/-1V
0x0104	+/-500mV
0x0103	+/-150mV
0x0182	0~20mA
0x0180	4~20mA

A.2.3 WISE-4250-S250

Table A.10: WISE-4250-S250							
Address 0X	**Ch**	**Description**	**Attribute**	**Address 4X**	**Ch**	**Description**	**Attribute**
00001	0	DI Value	Read	**40001~40002**	0	Counter/Frequency Value	Read
00002	1		Read	**40003~40004**	1		Read
00003	2		Read	**40005~40006**	2		Read
00004	3		Read	**40007~40008**	3		Read
00005	4		Read	**40009~40010**	4		Read
00006	5		Read	**40011~40012**	5		Read
00033	0	Counter Start(1)/ Stop(0)	R/W	**40301**	All	DI Value	Read
00034	1		R/W				
00035	2		R/W				
00036	3		R/W				
00037	4		R/W				
00038	5		R/W				
00039	0	Clear Counter(1)	Write				
00040	1		Write				
00041	2		Write				
00042	3		Write				
00043	4		Write				
00044	5		Write				

Table A.10: WISE-4250-S250							
00045	0	Clear Overflow	R/W				
00046	1		R/W				
00047	2		R/W				
00048	3		R/W				
00049	4		R/W				
00050	5		R/W				
00051	0	DI Low Latch Status	R/W				
00052	1		R/W				
00053	2		R/W				
00054	3		R/W				
00055	4		R/W				
00056	5		R/W				
00057	0	DI High Latch Status	R/W				
00058	1		R/W				
00059	2		R/W				
00060	3		R/W				
00061	4		R/W				
00062	5		R/W				
DO							
Address 0X	**Ch**	**Description**	**Attribute**	**Address 4X**	**Ch**	**Description**	**Attribute**
00017	0	DO Value	R/W	**40013~40014**	0	Pulse Output Low Level Width	R/W
00018	1		R/W	**40015~40016**	1		R/W
				40017~40018	0	Pulse Output High Level Width	R/W
				40019~40020	1		R/W
				40021~40022	0	Set Absolute Pulse	R/W
				40023~40024	1		R/W
				40025~40026	0	Set Incremental Pulse	R/W
				40027~40028	1		R/W
				40303	All	DO Value	R/W
Modbus RTU							
Address 0X	**Ch**	**Description**	**Attribute**	**Address 4X**	**Ch**	**Description**	**Attribute**
01001~01064	0~63	Bit value (COM 1)	R/W	**41001~41064**	0~63	Word value (COM1)	R/W
				41101~41164	0~63	Bit value error code (COM1)	Read
				41201~41264	0~63	Word value error code (COM1)	Read
Miscellaneous							
Address 0X	**Ch**	**Description**	**Attribute**	**Address 4X**	**Ch**	**Description**	**Attribute**
05001		RTC Battery Low	Read	**45101**		Data Logger Status	Read
				45302		RSSI	Read
Addresses for Internal Use							
Address 0X	**Ch**	**Description**	**Attribute**	**Address 4X**	**Ch**	**Description**	**Attribute**
				40211		Module Name 1	Read
				40212		Module Name 2	Read
				40213		Module Name 3	Read
				40214		Reserved for Module Name	Read

A.2.4 WISE-4250-S251

Table A.11: WISE-4250-S251							
Sensor (Max 10)	**0**	**DI (Max 12)**		**6**	**DO (Max 12)**		**0**
AI (Max 12)	**0**	**COM(MAX 2)**		**1**			
DI							
Address 0X	**Ch**	**Description**	**Attribute**	**Address 4X**	**Ch**	**Description**	**Attribute**
00001	0	DI Value	Read	**40001~40002**	0	Counter/Frequency Value	Read
00002	1		Read	**40003~40004**	1		Read
00003	2		Read	**40005~40006**	2		Read
00004	3		Read	**40007~40008**	3		Read
00005	4		Read	**40009~40010**	4		Read
00006	5		Read	**40011~40012**	5		Read
00033	0	Counter Start(1)/ Stop(0)	R/W	**40301**	All	DI Value	Read
00034	1		R/W				
00035	2		R/W				
00036	3		R/W				
00037	4		R/W				
00038	5		R/W				
00039	0	Clear Counter(1)	Write				
00040	1		Write				
00041	2		Write				
00042	3		Write				
00043	4		Write				
00044	5		Write				
00045	0	Clear Overflow	R/W				
00046	1		R/W				
00047	2		R/W				
00048	3		R/W				
00049	4		R/W				
00050	5		R/W				
00051	0	DI Low Latch Status	R/W				
00052	1		R/W				
00053	2		R/W				
00054	3		R/W				
00055	4		R/W				
00056	5		R/W				
00057	0	DI High Latch Status	R/W				
00058	1		R/W				
00059	2		R/W				
00060	3		R/W				
00061	4		R/W				
00062	5		R/W				
Modbus RTU							
Address 0X	**Ch**	**Description**	**Attribute**	**Address 4X**	**Ch**	**Description**	**Attribute**
01001~01064	0~63	Bit value (COM 1)	R/W	**41001~41064**	0~63	Word value (COM1)	R/W
				41101~41164	0~63	Bit value error code (COM1)	Read
				41201~41264	0~63	Word value error code (COM1)	Read

Table A.11: WISE-4250-S251							
Miscellaneous							
Address 0X	**Ch**	**Description**	**Attribute**	**Address 4X**	**Ch**	**Description**	**Attribute**
05001		RTC Battery Low	Read	**45101**		Data Logger Status	Read
				45302		RSSI	Read
Addresses for Internal Use							
Address 0X	**Ch**	**Description**	**Attribute**	**Address 4X**	**Ch**	**Description**	**Attribute**
				40211		Module Name 1	Read
				40212		Module Name 2	Read
				40213		Module Name 3	Read
				40214		Reserved for Module Name	Read

Appendix B

REST for WISE-4250 Series

B.1 Introduction

REST is the short for Representational State Transfer. It's a software architecture style widely used for creating scalable web services. It is web standards based architecture and uses HTTP Web service. APIs that adhere to the REST architectural constraints are called RESTful APIs.

HTTP-based RESTful APIs are defined with these aspects:

- Base URI, such as http://example.com/resources/
- An Internet media type for the data. This is often JSON but can be any other valid Internet media type (e.g., XML, Atom, microformats, images, etc.)
- Standard HTTP methods (e.g., GET, PUT, POST, or DELETE)

User could find the REST resources for WISE-4000 series in the appendix b of user manual.

Take the web-service that related to DI as an example, you could find it meets the definition of RESTful.

URL structure:

http://10.0.0.1/di_value/slot_index (Collection URI)

http://10.0.0.1/di_value/slot_index/ch_num (Element URI)

Internet media type:

application/json

```
{
 "Ch":2,
 "Md":1,
 "Stat":0,
 "Val":3378,
 "Cnting":0,
 "ClrCnt":1,
 "OvLch": 0
}
```

(Please note that ADAM-6000-CE and ADAM-6200 series use XML as the Internet media type)

HTTP methods:

GET:Returns the representation of all of digital input value resource.

PUT:Replace all of digital input value resource

PATCH:Apply partial modifications to digital input value resource.

B.2 REST Resources for WISE-4250 Series

B.2.1 Device Control

/control

Table B.1: Device control

Description	The system can be controlled by command objects.
URL Structure	**http://10.0.0.1/control**
HTTP Method	PATCH: Send the control command to module.
GET	None
PUT	None

PATCH	Request: **PATCH /control**
	
	[Example]:
	■ Request: **PATCH /control**, Reset the module to default settings and restart it.
	Content-type: application/json
	{
	"RFD":1
	}
	
	Response: 200 OK

■ Resource value definitions:

Field	Abbreviation	Data Type	Property	Description
System Restart	Rst	Number	W	1: Restart the system * Module will automatically restart.
Module Locate function	Lc	Number	W	1: Locate the module. 0: Stop locating the module
Reset to Factory Default	RFD	Number	W	1: Reset to factory default settings and then restart the system * User must restart module.
Reset to Calibration Default Value	RCD	Number	W	1: Reset to Calibration default coefficient
Reset Password	RPw	Number	W	1: Reset the system password * User must restart module.
Clear Log data	ClrLg	Number	W	1: Reset the built-in log data (IO data) 2: Reset the built-in log data (System diagnosis)
Change password	Pw	String	W	The new password. <ref to Note 1> Example, Set the password '00000000' from '123'
AI calibration	AIC	Number	W	0 or 1: AI calibration.

Account	
user	"73370F0F0F0F0F0F0F0F0F0E0D0C1F1F1F1F1F"
admin	"72370F0F0F0F0F0F0F0F0F0E0D0C1F1F1F1F1F"
root	"77370F0F0F0F0F0F0F0F0F0E0D0C1F1F1F1F1F"

0	Span Calibration
1	Zero Calibration

Reset Modbus RTU response status	RRS	Number	W	1: Reset all of Modbus RTU response status in COM1 2: Reset all of Modbus RTU response status in COM2

<Note 1> Password encoding
Only 'root' account has the right to change the passwords.

The encoding procedure:

1. Make the source data array, S, according to the order, Account Type (1 byte), password valid byte-Length (1 byte), New Password (8 bytes), and Current Password (8 bytes).

Account(1)	Len(1)	New Password (8)	Current Password (8)
user: 'L' admin: 'M' root: 'H'	Number (1 – 8)	New Password (Use 0x20 for un-used bytes.)	Current Password (Use 0x20 for un-used bytes.)

2. Perform a XOR(^) operation on array S and the constant value, 0x3F. The result is named R.
3. Convert each byte in R into two ASCII characters. The length of the configured password string will be 20.

For example, encode a new password string '123456' for admin account with the original password, '00000000'.

1. S will be [0x4D, 0x06, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x20, 0x20, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30].
2. XOR with 0x3F. Get R to be [0x72, 0x39, 0x0E,0x0D, 0x0C, 0x0B, 0x0A, 0x09, 0x1F, 0x1F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F].
3. The result string will be "72390E0D0C0B0A091F1F0F0F0F0F0F0F0F0F".

Remarks	

B.2.2 Digital Input

/di_config/slot_0

/di_config/slot_0/ch_num

Table B.2: Digital Input	
Description	Retrieves information about the digital input configuration resource.
URL Structure	**http://10.0.0.1/di_config/slot_macID** **http://10.0.0.1/di_config/slot_macID/ch_num** **where macID = xxyyzziijjkk: the MAC ID of WISE-4210 end device (node)** **= 0: for other models.** **where num = 0 ~ : the channel number**
HTTP Method	GET: Return the representation of all of digital input configurations resource. PUT: Replace all of digital input configurations resource. PATCH: Apply partial modifications to digital input configurations resource.
GET	Multi Channel Request: **GET /di_config/slot_macID** Single Channel Request: **GET /di_config/slot_macID/ch_num** [Example]: In case of WISE-4210 I/O node, such as WISE-4210-S250 with 6 DI channels Request: **GET /di_config/slot_AD4210112233** Content-type: application/json Response: 200 OK { "DICfg": [{ "Ch":0, "En":1, "Md":0, "Inv":0, "Fltr": 1, "FtLo": 100, "FtHi": 150, "FqT": 5, "FqP": 0, "CntIV": 0, "CntKp": 0, "Tag": "Digital Input 0" }, { "Ch":1, "En":1, "Md":0, "Inv":1, "Fltr": 0, "FtLo": 1000, "FtHi": 1000, "FqT": 10, "FqP": 0, "CntIV": 0, " CntKp": 0, "Tag": "Digital Input 1" },

GET (Cont.)	{
	"Ch":2,
	"En":1,
	"Md":0,
	"Inv":0,
	"Fltr": 1,
	"FtLo": 500,
	"FtHi": 500,
	"FqT": 10,
	"FqP": 0,
	"CntIV": 1000,
	" CntKp": 1,
	"Tag": "Input Counter"
	},
	{
	"Ch":3,
	"En":1,
	"Md":0,
	"Inv":0,
	"Fltr": 0,
	"FtLo": 100,
	"FtHi": 100,
	"FqT": 2,
	"FqP": 1,
	"CntIV": 0,
	" CntKp": 0,
	"Tag": "Input Frequency"
	},
	{
	"Ch":4,
	"En":1,
	"Md":255,
	"Inv":0,
	"Fltr": 1,
	"FtLo": 500,
	"FtHi": 500,
	"FqT": 10,
	"FqP": 0,
	"CntIV": 1000,
	" CntKp": 1,
	"Tag": "Input 5"
	},
	{
	"Ch":5,
	"En":1,
	"Md":255,
	"Inv":0,
	"Fltr": 0,
	"FtLo": 100,
	"FtHi": 100,
	"FqT": 2,
	"FqP": 1,
	"CntIV": 0,
	" CntKp": 0,
	"Tag": "Input 6"
	}
	]
	}

GET (Cont.)	■ Request : GET /di_config/slot_AD4210112233/ch_2
	
	Content-type: application/json
	Response: 200 OK
	{
	"Ch":2,
	"En":1,
	"Md":0,
	"Inv":1,
	"Fltr": 0,
	"FtLo": 10000,
	"FtHi": 10000,
	"FqT": 10,
	"FqP": 0,
	"CntIV": 0,
	" CntKp": 0,
	"Tag": "FrontDoor"
	}

PUT	Single/Multi Channel Request: **PUT /di_config/slot_macID** Single Channel Request: **PUT /di_config/slot_macID/ch_num** [Example]: ■ Request: **PUT /di_config/slot_AD4210112233** Content-type: application/json { "DICfg": [{ "Ch":0, "En":1, "Md":0, "Inv":0, "Fltr": 1, "FtLo": 100, "FtHi": 150, "FqT": 5, "FqP": 0, "CntIV": 0, " CntKp": 0, "Tag": "Digital Input 0" }, { "Ch":1, "En":1, "Md":0, "Inv":1, "Fltr": 0, "FtLo": 1000, "FtHi": 1000, "FqT": 10, "FqP": 0, "CntIV": 0, " CntKp": 0, "Tag": "Digital Input 1" },

PUT (Cont.)	{
	"Ch":3,
	"En":1,
	"Md":255,
	"Inv":0,
	"Fltr": 0,
	"FtLo": 100,
	"FtHi": 100,
	"FqT": 2,
	"FqP": 1,
	"CntIV": 0,
	" CntKp": 0,
	"Tag": "Input Frequency"
	},
	{
	"Ch":4,
	"En":1,
	"Md":255,
	"Inv":0,
	"Fltr": 1,
	"FtLo": 500,
	"FtHi": 500,
	"FqT": 10,
	"FqP": 0,
	"CntIV": 1000,
	" CntKp": 1,
	"Tag": "Input 5"
	},
	{
	"Ch":5,
	"En":1,
	"Md":255,
	"Inv":0,
	"Fltr": 0,
	"FtLo": 100,
	"FtHi": 100,
	"FqT": 2,
	"FqP": 1,
	"CntIV": 0,
	" CntKp": 0,
	"Tag": "Input 6"
	}
	]
	}
	
	Response: 200 OK

PUT (Cont.)	■ Request: **PUT /di_config/slot_AD4210112233/ch_3**
	Content-type: application/json
	{
	"Ch":3,
	"En":1,
	"Md":255,
	"Inv":0,
	"Fltr": 1,
	"FtLo": 5000,
	"FtHi": 5000,
	"FqT": 2,
	"FqP": 1,
	"CntIV": 0,
	
	" CntKp": 1,
	"Tag": "Station_A"
	}
	
	Response: 200 OK

PATCH	Single/Multi Channel Request:
	PATCH /di_config/slot_macID
	Single Channel Request:
	PATCH /di_config/slot_macID/ch_num
	
	[Example]:
	■ Request: **PATCH /di_config/slot_AD4210112233**
	Content-type: application/json
	{
	"DICfg": [
	{
	"Ch":0,
	"Inv":0
	},
	{
	"Ch":2,
	"CntKp": 0
	},
	{
	"Ch":3,
	"Md":0,
	"Fltr": 1,
	"FtLo": 500,
	"FtHi": 500
	}
	]
	}
	
	Response: 200 OK
	
	■ Request: **PATCH /di_config/slot_AD4210112233/ch_3**
	Content-type: application/json
	{
	"Ch":3,
	"Md":255
	}
	
	Response: 200 OK

■ JSON array name definition:

Field	Abbreviation	Data Type
Array of Digital input configurations	DICfg	Array

■ Resource value definitions:

Field	Abbreviation	Data Type	Property	Description
Channel Number	Ch	Number	R	0, 1, …: Digital input channel number.
Channel Enable	En	Number	RW	1 / 0: Enable / Disable function of this input channel.
Mode	Md	Number	RW	Digital input mode. 0 DI 1 Counter 2 LowToHighLatch 3 HighToLowLatch 4 Frequency 255 Invalid (Read only) *Frequency is not supported in low power mode.
Invert Signal	Inv	Number	RW	1 or 0: Enable or Disable invert signal function.
Digital Filter	Fltr	Number	RW	1 or 0: Enable or Disable digital filter function
Min. Low Signal Width	FtLo	Number	RW	Minimum low signal width of digit filter 1 – 4294967295 (0.1 ms) *UDI unit: 5000 (0.1ms), and round up to multiple of 5000 Ex: If value is 5001, the filter width will be 1000 ms
Min. High Signal Width	FtHi	Number	RW	Minimum high signal width of digit filter 1 – 4294967295 (0.1 ms). *UDI unit: 5000 (0.1ms), and round up to multiple of 5000 Ex: If value is 5001, the filter width will be 1000 ms
Frequency Value Reset (to zero) Time	FqT	Number	RW	The frequency measurement will be cleared to 0 if no input signal changed within this period of time. 1 – 255 (0.1 sec), default: 100 (i.e. 10 sec) (Taking 0.1 Hz frequency precise for example, it will take up to 10 seconds to verify the input signal is indeed removed. Users could configure this parameter to reduce the response time.) *No effect in UDI

Frequency Precise 0.01Hz	FqP	Number	RW	1 or 0: Enable / Disable Frequency Precise 0.01Hz (Frequency Precise is 0.01Hz or 0.1Hz. Default Frequency Precise is 0.1Hz) *No effect in UDI
Counter Startup Value	CntIV	Number	RW	User defined initial counter value 0 – 4294967295
Keep Counter Value When Poweroff	CntKp	Number	RW	1 or 0: Enable / Disable keep last value when power off.
Tag Name	Tag	String	RW	The description tag for this channel. Max. 21 characters
Power Output Delay time	DTim	Number	RW	1~1800, unit: s Default: 1
Power Output Selection	PE	Number	RW	Default: 0

0	Disable
1	Power Output 0
2	Power Output 1

DI conversion interval	CRT	Number	W	The interval for DI conversion for power saving 1~86400, unit: Sec (Default: 1) *No use for power output is disable.

Remarks	

B.2.3 DI Data Acquisition

/di_value/slot_0

/di_value/slot_0/ch_num

Table B.3: DI Data Acquisition	
Description	Retrieves information about the digital input value resource.
URL Structure	**http://10.0.0.1/di_value/slot_macID** **http://10.0.0.1/di_value/slot_macID/ch_num** **where macID = xxyyzziijjkk: the MAC ID of WISE-4210 end device (node)** **= 0: for other models.** **where num = 0 ~ : the channel number**
HTTP Method	GET: Returns the representation of all of digital input value resource. PUT: Replace all of digital input value resource PATCH: Apply partial modifications to digital input value resource.
GET	Multi Channel Request: **GET /di_value/slot_macID** Single Channel Request: **GET /di_value/slot_macID/ch_num** [Example]: In case of WISE-4210 I/O node, such as WISE-4210-S250 with 6 DI channels ■ Request: **GET /di_value/slot_AD4210112233** Content-type: application/json Response: 200 OK “DIVal”: [{ “Ch”:0, "En":1, "Md":0, “Stat”:1, “Cnting”:0, “OvLch”: 0, “CtFq” : 10, “Fq”: 100, “Lch”: 0, “Hch”: 0 },

GET (Cont.)	{ {
	“Ch”:1,
	"En":1,
	"Md":0,
	“Stat”:0,
	“Cnting”:0,
	“OvLch”: 0,
	“CtFq” : 10,
	“Fq”: 100,
	“Lch”: 0,
	“Hch”: 0
	},
	{
	“Ch”:2,
	"En":1,
	"Md":0,
	“Stat”:0,
	“Cnting”:1,
	“OvLch”: 0,
	“CtFq” : 10,
	“Fq”: 100,
	“Lch”: 0,
	“Hch”: 0
	},
	{
	“Ch”:3,
	"En":1,
	"Md":255,
	“Stat”:0,
	“Cnting”:0,
	“OvLch”: 0,
	“CtFq” : 0,
	“Fq”: 0,
	“Lch”: 0,
	“Hch”: 0
	},
	{
	“Ch”:4,
	"En":1,
	"Md":255,
	“Stat”:0,
	“Cnting”:0,
	“OvLch”: 0,
	“CtFq” : 0,
	“Fq”: 0,
	“Lch”: 0,
	“Hch”: 0
	},

GET (Cont.)	{
	“Ch”:5,
	"En":1,
	"Md":255,
	“Stat”:0,
	“Cnting”:0,
	“OvLch”: 0,
	“CtFq” : 0,
	“Fq”: 0,
	“Lch”: 0,
	“Hch”: 0
	}
	]
	}
	
	■ Request : GET /di_value/slot_AD4210112230/ch_2
	Content-type: application/json
	Response: 200 OK
	{
	“Ch”:2,
	"En":1,
	"Md":0,
	“Stat”:1,
	“Cnting”:0,
	“OvLch”: 0,
	“CtFq” : 10,
	“Fq”: 100,
	“Lch”: 0,
	“Hch”: 0
	}

PUT	Single/Multi Channel Request: **PUT /di_value/slot_macID** Single Channel Request: **PUT /di_value/slot_macID/ch_num** [Example]: In case of WISE-4210 I/O node, such as WISE-4210-S250 with 6 DI channels ■ Request: **PUT /di_value/slot_AD4210112200** Content-type: application/json { “DIVal”: [{ “Ch”:0, "En":1, "Md":0, “Stat”:0, “Cnting”:0, “ClrCnt”:0, “OvLch”: 0, “CtFq” : 0, “Fq”: 0, “Lch”: 0, “Hch”: 0 },

PUT (Cont.)	{ “Ch”:1, "En":1, "Md":0, “Stat”:0, “Cnting”:0, “ClrCnt”:1, “OvLch”: 0, “CtFq” : 0, “Fq”: 0, “Lch”: 0, “Hch”: 0 }, { “Ch”:2, "En":1, "Md":0, “Stat”:0, “Cnting”:0, “ClrCnt”:1, “OvLch”: 0, “CtFq” : 0, “Fq”: 0, “Lch”: 0, “Hch”: 0 }, { “Ch”:3, "En":1, "Md":0, “Stat”:0, “Cnting”:0, “ClrCnt”:0, “OvLch”: 0, “CtFq” : 0, “Fq”: 0, “Lch”: 0, “Hch”: 0 }, { “Ch”:4, "En":1, "Md":0, “Stat”:0, “Cnting”:0, “ClrCnt”:1, “OvLch”: 0, “CtFq” : 0, “Fq”: 0, “Lch”: 0, “Hch”: 0 },

PUT (Cont.)	{
	“Ch”:5,
	"En":1,
	"Md":0,
	“Stat”:0,
	“Cnting”:0,
	“ClrCnt”:0,
	“OvLch”: 0,
	“CtFq” : 0,
	“Fq”: 0,
	“Lch”: 0,
	“Hch”: 0
	}
	]
	}
	
	Response: 200 OK
	
	■ Request: **PUT /di_value/slot_AD4210112200/ch_2**
	Content-type: application/json
	{
	“Ch”:2,
	"En":1,
	"Md":0,
	“Stat”:0,
	“Cnting”:0,
	“ClrCnt”:1,
	“OvLch”: 0,
	“CtFq” : 0,
	“Fq”: 0,
	“Lch”: 0,
	“Hch”: 0
	}
	
	Response: 200 OK

PATCH

Single/Multi Channel Request:
PATCH /di_value/slot_macID
Single Channel Request:
PATCH /di_value/slot_macID/ch_num

[Example]:

- Request: **PATCH /di_value/slot_AD4210112200**

```
Content-type: application/json
{
“DIVal”: [
   {
     “Ch”:2,
     “Cnting”: 1
   },
   {
     “Ch”:3,
     “OvLch”:0
   }
 ]
}
```

Response: 200 OK

- Request: **PATCH /di_value/slot_AD4210112200/ch_3**

```
Content-type: application/json
{
“Ch”:3,
     “ClrCnt”:1
}
```

Response: 200 OK

- JSON array name definition:

Field	Abbreviation	Data Type
Array of Digital input configurations	DICfg	Array

- Resource value definitions:

Field	Abbreviation	Data Type	Property	Description
Channel Number	Ch	Number	R	0, 1, …: Digital input channel number.
Channel Enable	En	Number	RW	1 / 0: Enable / Disable function of this input channel.
Enable/Disable Channel	Md	Number	R	Enable/Disable DI/UDI channel

0	DI
1	Counter
2	LowToHighLatch
3	HighToLowLatch
4	Frequency
255	Invalid mode, when UDI disable

*Frequency mode is not supported in UDI module
*Frequency is not supported in low power mode.

Signal Logic Status	Stat	Number	R	1, 0: Input signal is Logic High or Low.
Start Counter	Cnting	Number	RW	Start/Stop counter counting Read 1: counter is counting 0: not counting Write 1: start counting 0: stop counting
Clear Counter	ClrCnt	Number	W	1: Clear the counter value
Get/Clear Counter Overflow	OvLch	Number	RW	counter overflow Read 1: overflow occurred. 0: no overflow Write 0: clear the overflow
Counter Value	CtFq	Number	R	The counter value
Frequency Value	Fq	Number	R	■ Value is multiplied by 10 if frequency precise is 0.1Hz ■ Value is multiplied by 100 if frequency precise is 0.01Hz *Frequency is not supported in low power mode. *WISE-4210-AP only support frequency precise is 0.01Hz.
Get/Clear L2H Latch Status	Lch	Number	RW	L2H latch status Read 1: L2H latch occurred. 0: no L2H latch Write 0: clear the L2H latch status
Get/Clear H2L Latch Status	Hch	Number	RW	H2L latch status Read 1: H2L latch occurred. 0: no H2L latch Write 0: clear the H2L latch status
Channel Event Status	Evt	Number	R	DI statuses

Bit Order	Description
0	DI Not Ready
1~7	Reserved

Remarks

B.2.4 Digital Output

/do_config/slot_0

/do_config/slot_0/ch_num

Table B.4: Digital Output	
Description	Retrieves information about the digital output configuration resource on specific slot.
URL Structure	**http://10.0.0.1/do_config/slot_macID** **http://10.0.0.1/do_config/slot_macID/ch_num** **where macID = xxyyzziijjkk: the MAC ID of WISE-4210 end device (node)** **= 0: for other models.** **where num = 0 ~ : the channel number**
HTTP Method	GET: Return the representation of all of digital output configurations resource. PUT: Replace all of digital output configurations resource. PATCH: Apply partial modifications to digital output configurations resource.
GET	Multi Channel Request: **GET /do_config/slot_macID** Single Channel Request: **GET /do_config/slot_macID/ch_num** [Example]: ■ Request: **GET /do_config/slot_0** Content-type: application/json Response: 200 OK { "DOCfg": [{ "Ch":0, "En":1, "Md":0, "FSV":0, "PsLo":0, "PsHi":0, "HDT": 0, "LDT": 0, "ACh": 0, "AMd": 0, "Tag": "Digital Output 0" }, { "Ch":1, "En":1, "Md":3, "FSV":0, "PsLo":0, "PsHi":0, "HDT": 20000, "LDT": 0, "ACh": 0, "AMd": 0, "Tag": "Digital Output 1" },

GET (Cont.)	{
	"Ch":2,
	"En":1,
	"Md":1,
	"FSV":0,
	"PsLo":5000,
	"PsHi":15000,
	"HDT": 0,
	"LDT": 0,
	"ACh": 0,
	"AMd": 0,
	"Tag": "Pulse"
	},
	{
	"Ch":3,
	"En":1,
	"Md":2,
	"FSV":0,
	"PsLo":0,
	"PsHi":0,
	"HDT": 0,
	"LDT": 3000,
	"ACh": 0,
	"AMd": 0,
	"Tag": "Low2High"
	}
	]
	}
	
	■ Request: **GET /do_config/slot_AD4210112233/ch_9**
	
	Content-type: application/json
	Response: 200 OK
	{
	"Ch":9,
	"En":1,
	"Md":0,
	"FSV":0,
	"PsLo":0,
	"PsHi":0,
	"HDT": 0,
	"LDT": 0,
	"ACh": 0,
	"AMd": 0,
	"Tag": "DigOut"
	}

PUT	Single/Multi Channel Request: PUT /do_config/slot_macID Single Channel Request: PUT /do_config/slot_macID/ch_num [Example]: ■ Request: **PUT /do_config/slot_0** Content-type: application/json { "DOCfg": [{ "Ch":0, "En":1, "Md":0, "FSV":0, "PsLo":0, "PsHi":0, "HDT": 0, "LDT": 0, "ACh": 0, "AMd": 0, "Tag": "DOutput 0" }, { "Ch":1, "En":1, "Md":3, "FSV":0, "PsLo":0, "PsHi":0, "HDT": 20000, "LDT": 0, "ACh": 0, "AMd": 0, "Tag": "Digital Output 1" }, { "Ch":2, "En":1, "Md":1, "FSV":0, "PsLo":5000, "PsHi":15000, "HDT": 0, "LDT": 0, "ACh": 0, "AMd": 0, "Tag": "Pulse" },

PUT (Cont.)	{
	"Ch":3,
	"En":1,
	"Md":2,
	"FSV":0,
	"PsLo":0,
	"PsHi":0,
	"HDT": 0,
	"LDT": 3000,
	"ACh": 0,
	"AMd": 0,
	"Tag": "Low2High"
	}
	]
	}
	Response: 200 OK
	
	■ Request: **PUT /do_config/slot_0/ch_3**
	Content-type: application/json
	{
	"Ch":3,
	"En":1,
	"Md":1,
	"FSV":1,
	"PsLo":3000,
	"PsHi":7000,
	"HDT": 0,
	"LDT": 0,
	"ACh": 0,
	"AMd": 0,
	"Tag": "Station_A"
	}
	
	Response: 200 OK

PATCH

Single/Multi Channel Request:
PATCH /do_config/slot_ macID
Single Channel Request:
PATCH /do_config/slot_macID/ch_num

[Example]:

■ Request: **PATCH /do_config/slot_0**

```
Content-type: application/json
{
"DOCfg": [
   {
     "Ch":0,
     "Md":1
   },
   {
     "Ch":2,
     "FSV": 0
   }
 ]
}
```

Response: 200 OK

■ Request: **PATCH /do_config/slot_0/ch_3**

```
Content-type: application/json
{
"Ch":3,
     "Md":0
}
```

Response: 200 OK

■ JSON array name definition:

Field	Abbreviation	Data Type
Array of Digital input configurations	DOCfg	Array

■ Resource value definitions:

Field	Abbreviation	Data Type	Property	Description
Channel Number	Ch	Number	R	0, 1, …: Digital output channel number.
Channel Enable	En	Number	RW	1 / 0: Enable / Disable function of this output channel.
Mode	Md	Number	R	Digital output mode.
Fail Safety Value	FSV	Number	RW	1 or 0: Enable or Disable fail safety value function.
Low Signal Width	PsLo	Number	RW	Low signal width of pulse 1 - 65535 (0.1 ms)
High Signal Width	PsHi	Number	RW	High signal width of pulse 1 - 65535 (0.1 ms).

Value	Mode	Note
0	DO	
1	Pulse Output	
2	LowToHighDelay	
3	HighToLowDelay	
4	AI Alarm Drived	ADAM-T110. ADAM-3600

High To Low Delay Time	HDT	Number	RW	Time for High To Low Delay 1 - 65535 (0.1 ms)
Low To High Delay Time	LDT	Number	RW	Time for Low To High Delay 1 - 65535 (0.1 ms).
Tag Name	Tag	String	RW	The description tag for this channel. Max. 21 characters
The Number of AI Channel which drives the DO signal	ACh	Number	RW	0, 1, …: Analog input channel number to drive the DO
The Driving Alarm Mode	AMd	Number	RW	The driving mode

0	No
1	High alarm drived
2	Low alarm drived

Remarks	

B.2.5 DO Data Acquisition

/do_value/slot_0

/do_ value/slot_0/ch_num

Table B.5: DO Data Acquisition	
Description	Retrieves information about the digital input value resource on specific slot.
URL Structure	**http://10.0.0.1/do_value/slot_macID** **http://10.0.0.1/do_value/slot_macID/ch_num** **where index = 0 : the core module** **1 ~ : the identifier of I/O extension slot** **where num = 0 ~ : the channel number**
HTTP Method	GET: Returns the representation of all of digital output value resource. PUT: Replace all of digital output value resource PATCH: Apply partial modifications to digital output value resource.
GET	GETMulti Channel Request: GET /do_value/slot_macID Single Channel Request: GET /do_value/slot_macID/ch_num [Example]: ?Request : GET /do_value/slot_0 Content-type: application/json Response: 200 OK { "DOVal": [{ "Ch":0, "En":1, "Md":0, "Stat":1, "Val":1, "PsCtn":0,

GET (Cont.)	"PsStop":0,
	"PsIV": 0
	},
	{
	"Ch":1,
	"En":1,
	"Md":0,
	"Stat":0,
	"Val":0,
	"PsCtn":0,
	"PsStop":0,
	"PsIV": 0
	},
	{
	"Ch":2,
	"En":1,
	"Md":1,
	"Stat":1,
	"Val":3378,
	"PsCtn":0,
	"PsStop":0,
	"PsIV": 0
	},
	{
	"Ch":3,
	"En":1,
	"Md":3,
	"Stat":1,
	"Val":1,
	"PsCtn":0,
	"PsStop":0,
	"PsIV": 0
	}
	]
	}
	
	■ Request: **GET /do_value/slot_0/ch_2**
	
	Content-type: application/json
	Response: 200 OK
	{
	"Ch":2,
	"En":1,
	"Md":0,
	"Stat":1,
	"Val":1,
	"PsCtn":0,
	"PsStop":0,
	"PsIV": 0
	}

PUT	Single/Multi Channel Request: **PUT /do_value/slot_macID** Single Channel Request: **PUT /do_value/slot_macID/ch_num** [Example]: ■ Request: **PUT /do_value/slot_0** Content-type: application/json

PUT (Cont.)	{
	"DOVal": [
	{
	"Ch":0,
	"En":1,
	"Md":0,
	"Stat":1,
	"Val":1,
	"PsCtn":0,
	"PsStop":0,
	"PsIV": 0
	},
	{
	"Ch":1,
	"En":1,
	"Md":0,
	"Stat":0,
	"Val":0,
	"PsCtn":0,
	"PsStop":0,
	"PsIV": 0
	},
	{
	"Ch":2,
	"En":1,
	"Md":1,
	"Stat":1,
	"Val":3378,
	"PsCtn":0,
	"PsStop":0,
	"PsIV": 0
	},
	{
	"Ch":3,
	"En":1,
	"Md":3,
	"Stat":1,
	"Val":1,
	"PsCtn":0,
	"PsStop":0,
	"PsIV": 0
	}
	]
	}
	
	Response: 200 OK
	
	■ Request: P**UT /do_value/slot_0/ch_2**
	Content-type: application/json
	{
	"Ch":2,
	"En":1,
	"Md":2,
	"Stat":0,
	"Val":0,
	"PsCtn":0,
	"PsStop":0,
	"PsIV": 0
	}
	
	Response: 200 OK

PATCH

Single/Multi Channel Request:
PATCH /do_value/slot_macID
Single Channel Request:
PATCH /do_value/slot_macID/ch_num

[Example]:

■ Request: **PATCH /do_value/slot_0**

```
Content-type: application/json
{
"DOVal": [
   {
    "Ch":2,
    "Md": 2
   },
   {
    "Ch":3,
    "PsStop":1
   }
 ]
}
```

Response: 200 OK

■ Request: **PATCH /do_value/slot_0/ch_3**

```
Content-type: application/json
{
"Ch":3,
    "PsCtn":1
}
```

Response: 200 OK

■ JSON array name definition:

Field	Abbreviation	Data Type
Array of Digital input configurations	DOVal	Array

■ Resource value definitions:

Field	Abbreviation	Data Type	Property	Description
Channel Number	Ch	Number	R	0, 1, …: Digital output channel number.
Channel Enable	En	Number	R	1 / 0: Enable / Disable function of this output channel.
Mode	Md	Number	R	Digital output mode. 1 Pulse Output 2 LowToHighDelay 3 HighToLowDelay
Signal Logic Status	Stat	Number	R	1, 0: Output signal is Logic High or Low.
Channel Value	Val	Number	RW	DO measurement data **Output Mode** / **Value Description** DO: Get the current signal status or set its status Pulse Output: Get or set the absolute pulse count value LowToHighDelay: Get the current signal status or set its status

				HighToLowDelay: Get the current signal status or set its status
Pulse Output Continue State	PsCtn	Number	RW	1 / 0: Pulse outputting is continuous or not.
Stop Pulse Output	PsStop	Number	W	1: Stop the pulse outputting. (Continue is disabled, Absolute and incremental values are reset to zero. DO signal status is set to logic low.)
Incremental Pulse Output Value	PsIV	Number	RW	Incremental Pulse Output Value
Remarks				

B.2.6 Analog Input

/ai_genconfig/slot_index

Table B.6: Analog Input	
Description	Retrieves information about the analog input configuration resource on specific slot.
URL Structure	**http://10.0.0.1/ai_genconfig/slot_index** **where index = 0 : the core module** **1 ~ : the identifier of I/O extension slot** **For WISE-4210 Sub-1G nodes, index is macID of node.** **index = xxyyzziijjkk: the MAC ID of end device (node)**
HTTP Method	GET: Return the representation of all of analog input configurations resource. PUT: Replace all of analog input configurations resource. PATCH: Apply partial modifications to analog input configurations resource.

GET	Multi Channel Request:
	GET /ai_genconfig/slot_index
	
	[Example]:
	■ Request: **GET /ai_genconfig/slot_1**
	
	Content-type: application/json
	Response: 200 OK
	{
	"Res":16,
	"EnB":1,
	"BMd":0,
	"AiT":255,
	"Smp":1,
	"AvgM":5,
	"Inv":1
	}

PUT	Single/Multi Channel Request: **PUT /ai_genconfig/slot_index** [Example]: ■ Request: **PUT /ai_genconfig/slot_0** Content-type: application/json { "Res":16, "EnB":1, "BMd":0,

PUT (Cont.)

```
  "AiT":255,
  "Smp":1,
  "AvgM":0,
  "Inv":1
}

Response: 200 OK
```

PATCH

Single/Multi Channel Request:
PATCH /ai_genconfig/slot_index
[Example]:
■ Request: **PATCH /ai_genconfig/slot_0**

```
Content-type: application/json
{
  "EnB":0,
  "Smp":0
}

Response: 200 OK
```

■ Resource value definitions for all channels:

Field	Abbreviation	Data Type	Property	Description
AI Resolution	Res	Number	R	1 ~ 32: Number of bits for AI value. For example, the resolution of ADAM-T110 AI is 12-bit.
Burn-out Detection Enable	EnB	Number	RW	1 / 0: Enable / Disable AI burn-out detection
Burn-out Up/Down Scale Mode	BMd	Number	RW	The burn-out value 0: Down scale 1: Up scale
AI Filter Mode	AiT	Number	R	The AI filter mode 0: Auto (50/60Hz) 1: 50 Hz 2: 60 Hz 255: None
Channel Mask for AI Average	AvgM	Number	RW	The channel mask for average value.
AI conversion interval	Inv	Number	RW	The interval for AI conversion for power saving 1~86400, unit: Sec (Default: 1) *No use for performance mode
Allow two power ports output simultaneously	Alt	Number	RW	Default: 0 0: Disable 1: Enable

Remarks

B.2.7 AI Configuration

/ai_config/slot_0

/ai_config/slot_0/ch_num

Table B.7: AI Configuration	
Description	Retrieves information about the digital input configuration resource on specific slot.
URL Structure	**http://10.0.0.1/ai_config/slot_macID** **http://10.0.0.1/ai_config/slot_macID/ch_num** **where macID = xxyyzziijjkk: the MAC ID of WISE-4210 end device (node)** **= 0: for other models.** **where num = 0 ~ : the channel number**
HTTP Method	GET: Return the representation of all of analog input configurations resource. PUT: Replace all of analog input configurations resource. PATCH: Apply partial modifications to analog input configurations resource.
GET	Multi Channel Request: **GET /ai_config/slot_macID** Single Channel Request: **GET /ai_config/slot_macID/ch_num** [Example]: ■ Request: **GET /ai_config/slot_BF1234567899** Content-type: application/json Response: 200 OK { "AICfg": [{ "Ch":0, "En":1, "Rng":328, "EnLA": 1, "EnHA": 0, "LAMd": 0, "HAMd": 0, "LoA": "2.0", "HiA": "6.3", "LoS": "0.0", "HiS": "10", "Tag": "Analog Input 0" "LoP": "10.5", "HiP": "110.3" },

GET (Cont.)


```
    },
    {
      "Ch":2,
      "En":1,
      "Rng":328,
      "EnLA": 0,
      "EnHA": 0,
      "LAMd": 0,
      "HAMd": 0,
      "LoA": "0.0",
      "HiA": "0.0",
      "LoS": "0.0",
      "HiS": "10",
      "Tag": "AI2"
      "LoP": "10.5",
      "HiP": "110.3",
    },
    {
      "Ch":3,
      "En":0,
      "Rng":328,
      "EnLA": 0,
      "EnHA": 1,
      "LAMd": 0,
      "HAMd": 0,
      "LoA": "0.0",
      "HiA": "0.0",
      "LoS": "0.0",
      "HiS": "10",
      "Tag": "Analog"
      "LoP": "10.5",
      "HiP": "110.3",
    },
    {
      "Ch":4,
      "En":1,
      "Rng":328,
      "EnLA": 0,
      "EnHA": 0,
      "LAMd": 0,
      "HAMd": 0,
      "LoA": "0.0",
      "HiA": "0.0",
      "LoS": "0.0",
      "HiS": "10",
      "Tag": "Average AI"
      "LoP": "10.5",
      "HiP": "110.3",
    }
  ]
}
```

GET (Cont.)	■ Request: **GET /ai_config/slot_AD4210112233/ch_0**
	
	Content-type: application/json
	Response: 200 OK
	{
	"Ch":0,
	"En":1,
	"Rng":328,
	"EnA": 1,
	"AMd": 0,
	"LoA": "2.0",
	"HiA": "6.3",
	"LoS": "0.0",
	"HiS": "10",
	"Tag": "Analog Input 0"
	"LoP": "10.5",
	"HiP": "110.3",
	}

PUT	Single/Multi Channel Request: PUT /ai_config/slot_macID Single Channel Request: PUT /ai_config/slot_macID/ch_num [Example]: ■ Request: **PUT /ai_config/slot_AD4210112233** Content-type: application/json { "AICfg": [{ "Ch":0, "En":1, "Rng":328, "EnLA": 1, "EnHA": 1, "LAMd": 0, "HAMd": 0, "LoA": "2.0", "HiA": "6.3", "LoS": "4.0", "HiS": "6", "Tag": "Analog Input 0" "LoP": "10.5", "HiP": "110.3", },

PUT (Cont.)	{ "Ch":1, "En":0, "Rng":328, "EnLA": 0, "EnHA": 0, "LAMd": 0, "HAMd": 0, "LoA": "0.0", "HiA": "0.0", "LoS": "2", "HiS": "10", "Tag": "Analog Input 1" "LoP": "10.5", "HiP": "110.3", }, { "Ch":2, "En":1, "Rng":328, "EnLA": 0, "EnHA": 1, "LAMd": 0, "HAMd": 0, "LoA": "0.0", "HiA": "0.0", "Tag": "AI2" }, { "Ch":3, "En":0, "Rng":328, "EnLA": 0, "EnHA": 0, "LAMd": 0, "HAMd": 0, "LoA": "0.0", "HiA": "0.0", "LoS": "5", "HiS": "10", "Tag": "Analog" "LoP": "10.5", "HiP": "110.3", },

PUT (Cont.)	{
	"Ch":4,
	"En":0,
	"Rng":328,
	"EnLA": 0,
	"EnHA": 0,
	"LAMd": 0,
	"HAMd": 0,
	"LoA": "0.0",
	"HiA": "0.0",
	"LoS": "0",
	"HiS": "10",
	"Tag": "Analog"
	"LoP": "10.5",
	"HiP": "110.3",
	}
	]
	}
	
	Response: 200 OK
	
	■ Request: **PUT /ai_config/slot_AD4210112233/ch_3**
	Content-type: application/json
	{
	"Ch":3,
	"En":0,
	"Rng":328,
	"EnLA": 0,
	"EnHA": 0,
	"LAMd": 0,
	"HAMd": 0,
	"LoA": "0.0",
	"HiA": "0.0",
	"LoS": "4",
	"HiS": "5",
	"Tag": "Station_A"
	"LoP": "2.5",
	"HiP": "30.3",
	}
	
	Response: 200 OK

PATCH

Single/Multi Channel Request:
PATCH /ai_config/slot_macID
Single Channel Request:
PATCH /ai_config/slot_macID/ch_num

[Example]:

- Request: **PATCH /ai_config/slot_AD4210112233**

```
Content-type: application/json
{
"AICfg": [
   {
     "Ch":0,
     "LoA": "3.0"
   },
   {
     "Ch":2,
     "LoA": "2.0",
     "HiA": "8.0"
   }
 ]
}
```

Response: 200 OK

- Request: **PATCH /ai_config/slot_AD4210112233/ch_3**

```
Content-type: application/json
{
"Ch":3,
     "Tag": "Station_A12"
}
```

Response: 200 OK

- JSON array name definition:

Field	Abbreviation	Data Type
Array of Analog input configurations	AICfg	Array

- Resource value definitions by Each Channel (Total channels = AI channel number + 1 average channel):

Field	Abbreviation	Data Type	Property	Description
Channel Number	Ch	Number	R	0, 1, …: Analog input channel number.
Channel Enable	En	Number	RW	1 / 0: Enable / Disable AI conversion Notice: Average channel is read only. When channel mask of average is not 0, the value is 1.
Input Range	Rng	Number	RW	Analog input range. Notice: Average channel is read only.

Range code	
328	(0x0148) 0 – 10 V
327	(0x0147) 0 ~ 5 V
325	(0x0145) 0 ~ 1 V
262	(0x0106) 0 ~ 500 mV
261	(0x0105) 0 ~ 150 mV
323	(0x0143) +/- 10 V
322	(0x0142) +/- 5 V

				321 (0x0141) +/- 2.5 V 320 (0x0140) +/- 1 V 260 (0x0104) +/- 500 mV 384 (0x0180) 4 ~ 20 mA 385 (0x0181) +/- 20 mA 386 (0x0182) 0 ~ 20 mA 480 (0x01E0) DI (when channel in DI mode) 937 (0x03A9) PT100(385) -200~ +600ºC 969 (0x03C9) PT100(392) -200~ +600ºC 994 (0x03E2) PT1000 -40~ +160ºC 65535 Invalid range, when average channel is disable
Enable Low Alarm	EnLA	Number	RW	1 / 0: Enable / Disable AI low alarm function
Enable High Alarm	EnHA	Number	RW	1 / 0: Enable / Disable AI high alarm function
Alarm Low Mode	LAMd	Number	RW	The alarm mode 0 Momentary 1 Latch
Alarm High Mode	HAMd	Number	RW	The alarm mode 0 Momentary 1 Latch
Low Alarm Value	LoA	String	RW	Set/get the low alarm limit value. Data format is “±xxxxxxx.yyy” Value range: -2147483.647 ~ +2147483.647 For example, “+0003.250” or “3.25”
High Alarm Value	HiA	String	RW	Set/get the high alarm limit value. Data format is “±xxxxxxx.yyy” Value range: -2147483.647 ~ +2147483.647 For example, “+0015.250” or “15.25”
AI Min Scaling Value	LoS	String	RW	Set/get the scaling min value Data format is “±xxxxxxx.yyy” Value range: -2147483.647 ~ +2147483.647 For example, “+0004.350” or “4.35”
AI Max Scaling Value	HiS	String	RW	Set/get the scaling max value Data format is “±xxxxxxx.yyy” Value range: -2147483.647 ~ +2147483.647 For example, “+0016.720” or “16.72”
Tag Name	Tag	String	RW	The description tag for this channel. Max. 21 characters
Physical value for Min scaling	LoP	String	RW	Set/get the min scaling physical value Data format is “±xxxxxxx.yyy” Value range: -2147483.647 ~ +2147483.647 For example, “+0004.350” or “4.35”

Physical value for Max scaling	HiP	String		Set/get the max scaling physical value Data format is “±xxxxxxx.yyy” Value range: -2147483.647 ~ +2147483.647 For example, “+0120.350” or “120.35”
Power Output Selection	PE	Number	RW	Default: 0 0 Disable 1 Power Output 0 2 Power Output 1
Power Output Delay time	DTim	Number	RW	1~1800, unit: s Default: 5
Remarks				

B.2.8 AI Data Acquisition

/ai_value/slot_0

/ai_value/slot_0/ch_num

Table B.8: AI Data Acquisition

Description	Retrieves information about the analog input value resource on specific slot.
URL Structure	**http://10.0.0.1/ai_value/slot_macID** **http://10.0.0.1/ai_value/slot_macID/ch_num** **where macID = xxyyzziijjkk: the MAC ID of WISE-4210 end device (node)** **= 0: for other models.** **where num = 0 ~ : the channel number**
HTTP Method	GET: Returns the representation of all of analog input value resource. PUT: None PATCH: Apply partial modifications to analog input value resource.
GET	Multi Channel Request: GET /ai_value/slot_macID Single Channel Request: GET /ai_value/slot_macID/ch_num [Example]: ■ Request: **GET /ai_value/slot_BF1234567899** Content-type: application/json Response: 200 OK { "AIVal": [{ "Ch":0, "En":1, "Rng":328, "Val":148, "Eg":650, "Evt":0, "LoA": 0, "HiA": 0, "HVal":190, "HEg":1250,

GET (Cont.)	"LVal":15, "LEg":500, "SVal":148, "ClrH": 0, "ClrL": 0, “PEg”: 6123, "EgF":0.650, "HEgF":1.250, "LEgF":0.500, "PEgF":6.123, }, { "Ch":1, "En":1, "Rng":328, "Val":0, "Eg":0, "Evt":0, "LoA":0, "HiA":0, "HVal":0, "HEg":0, "LVal":0, "LEg":0, "SVal":0, "ClrH": 0, "ClrL": 0 “PEg”: 2000, "EgF":0.000, "HEgF":0.000, "LEgF":0.000, "PEgF":2.000, }, { "Ch":2, "En":1, "Rng":328, "Val":0, "Eg":0, "Evt":8, "LoA":0, "HiA":0, "HVal":0, "HEg":0, "LVal":0, "LEg":0, "SVal":0, "ClrH": 0, "ClrL": 0 “PEg”: 2000, "EgF":0.000, "HEgF":0.000, "LEgF":0.000, "PEgF":2.000, }, { "Ch":3, "En":1, "Rng":328, "Val":0,

GET (Cont.)


```
    "Eg":0,
    "Evt":0,
    "LoA":0,
    "HiA":0,
    "HVal":0,
    "HEg":0,
    "LVal":0,
    "LEg":0,
    "SVal":0,
    "ClrH": 0,
    "ClrL": 0
    “PEg”: 2000,
    "EgF":0.000,
    "HEgF":0.000,
    "LEgF":0.000,
    "PEgF":2.000,
   },
   {
    "Ch":4,
    "En":1,
    "Rng":328,
    "Val":0,
    "Eg":0,
    "Evt":0,
    "LoA":0,
    "HiA":0,
    "HVal":0,
    "HEg":0,
    "LVal":0,
    "LEg":0,
    "SVal":0,
    "ClrH": 0,
    "ClrL": 0
    “PEg”: 2000,
    "EgF":0.000,
    "HEgF":0.000,
    "LEgF":0.000,
    "PEgF":2.000,
   }
  ]
}
```


■ Request: **GET /ai_value/slot_BF1234567809/ch_2**


```
Content-type: application/json
Response: 200 OK
{
    "Ch":2,
"En":1,
    "Rng":328,
    "Val":0,
    "Eg":0,
    "Evt":8,
    "LoA":0,
    "HiA":0,
    "HVal":0,
    "HEg":0,
    "LVal":0,
    "LEg":0,
    "SVal":0,
```

<table>
<tr><td>GET (Cont.)</td><td>"ClrH": 0,
"ClrL": 0
"PEg": 2000,
"EgF":0.000,
"HEgF":0.000,
"LEgF":0.000,
"PEgF":2.000,
}</td></tr>
<tr><td>PUT</td><td>None</td></tr>
<tr><td>PATCH</td><td>Single/Multi Channel Request:
PATCH /ai_value/slot_macID
Single Channel Request:
PATCH /ai_value/slot_macID/ch_num

[Example]:
■ Request: PATCH /ai_value/slot_BF1234567890
Content-type: application/json
{
"AIVal": [
{
"Ch":2,
"LoA": 0
},
{
"Ch":3,
"HiA":0
}
]
}

Response: 200 OK

■ Request: PATCH /ai_value/slot_BF1234567899/ch_3
Content-type: application/json
{
"LoA":0
}

Response: 200 OK</td></tr>
</table>

■ JSON array name definition:

Field	Abbreviation	Data Type
Array of Analog input value	AIVal	Array

■ Resource value definitions (Total channels = AI channel number + 1 average channel):

Field	Abbreviation	Data Type	Property	Description
Channel Number	Ch	Number	R	0, 1, …: Analog input channel number. Note for the average channel: The average channel number for a 4-ch AI module is 4.
Input Range	Rng	Number	R	Analog input range.

Range code	
328	(0x0148) 0 – 10 V
327	(0x0147) 0 ~ 5 V
325	(0x0145) 0 ~ 1 V

				262 (0x0106) 0 ~ 500 mV 261 (0x0105) 0 ~ 150 mV 323 (0x0143) +/- 10 V 322 (0x0142) +/- 5 V 321 (0x0141) +/- 2.5 V 320 (0x0140) +/- 1 V 260 (0x0104) +/- 500 mV 384 (0x0180) 4 ~ 20 mA 385 (0x0181) +/- 20 mA 386 (0x0182) 0 ~ 20 mA 480 (0x01E0) UDI (when channel in UDI mode) 937 (0x03A9) PT100(385) -200~ +600ºC 969 (0x03C9) PT100(392) -200~ +600ºC 994 (0x03E2) PT1000 -40~ +160ºC 65535 Invalid range, when average channel is disable
Channel Enable	En	Number	R	1 / 0: Enable / Disable AI conversion Notice: Average channel is read only. When channel mask of average is not 0, the value is 1.
Channel Raw Value	Val	Number	R	0 ~ 65535: AI measurement data (Raw data)
Channel Event Status	Evt	Number	R	AI statuses Bit Order: Description 0: Fail to provide AI value (UART timeout, ADC error) 1: Over Range 2: Under Range 3: Open Circuit (Burnout) 4: AI Not Ready 5: Unavailable Channel Configuration (Channel Disabled, DI Mode Used) 6: Reserved 7: ADC initializing/Error 8: Reserved 9: Zero/Span Calibration Error 10~31: Reserved
Low Alarm Value	LoA	String	RW	Low alarm status Read 1 : low alarm occurred. 0 : not occurred Write 0 : clear the low alarm status

High Alarm Value	HiA	String	RW	High alarm status Read 1 : high alarm occurred. 0 : not occurred Write 0 : clear the high alarm status
Maximum AI Raw Value	HVal	Number	R	AI max. measurement data (Raw data)
Minimum AI Raw Value	LVal	Number	R	AI min. measurement data (Raw data)
Channel Raw Value After Scaling	SVal	Number	R	0 ~ 65535: AI measurement data (Raw data) after scaling
Clear Maximum AI Value	ClrH	Number	W	1: Clear the Maximum AI value
Clear Minimum AI Value	ClrL	Number	W	1: Clear the Minimum AI value
Channel Engineering data (floating type)	EgF	Number	R	AI engineering data, the value is floating type. Unit: mV or mA For example, 999.120 -> 999.12 mV -3.220-> -3.22 mA
Maximum AI Engineering data (floating type)	HEgF	Number	R	AI max. engineering data, the value is is floating type. Unit: mV or mA For example, 10.200 -> 10.2 mV -5.120 -> -5.12 mV
Minimum AI Engineering data (floating type)	LEgF	Number	R	AI min. engineering data, the value is floating type. Unit: mV or mA For example, 250.350 -> 250.35 mV -0.120-> -0.12 mV
Physical value after scaling (floating type)	PEgF	Number	R	AI physical value after scaling, the value is floating type For example: 150.350
Remarks				

B.2.9 Sensor Input

/sensor _config/slot_0/ch_num

Table B.9: Sensor input	
Description	Retrieves information about the sensor input configuration resource on specific slot.
URL Structure	**http://10.0.0.1/sensor_config/slot_macID** **http://10.0.0.1/sensor_config/slot_macID/ch_num** **where macID = xxyyzziijjkk: the MAC ID of WISE-4210 end device (node).** **= 0: for other models.** **where num = 0 ~ : the channel number**
HTTP Method	GET: Return the representation of all of sensor input configurations resource. PUT: Replace all of sensor input configurations resource. PATCH: Apply partial modifications to sensor input configurations resource.
GET	Multi Channel Request: **GET /sensor_config/slot_index** Single Channel Request: **GET /sensor_config/slot_index/ch_num** [Example]: ■ Request: **GET /sensor_config/slot_0** Content-type: application/json Response: 200 OK { "SCfg": [{ "Ch":0, "En":1, "Rng":4096, "EnLA": 1, "EnHA": 0, "LAMd": 0, "HAMd": 0, "LoA": "2.0", "HiA": "6.3", “Dev”: “10.5” “PIvt”: 1500, “Val”: “0”, “Tag": "W4220_CC0001-Temp" }, { "Ch":1, "En":1, "Rng":4128, "EnLA": 1, "EnHA": 1, "LAMd": 0, "HAMd": 0, "LoA": "0.0", "HiA": "10.0", “Dev”: “10.5” “PIvt”: 1500, “Val”: “1.0”, "Tag": " W4220_CC0001-RH" },

GET (Cont.)	{
	"Ch":2,
	"En":1,
	"Rng":4160,
	"EnLA": 0,
	"EnHA": 0,
	"LAMd": 0,
	"HAMd": 0,
	"LoA": "0.0",
	"HiA": "0.0",
	“Dev”: “10.5”
	“PIvt”: 1500,
	“Val”: “1.0”,
	"Tag": "W4220_CC0001-Press"
	},
	{
	"Ch":3,
	"En":1,
	"Rng":4192,
	"EnLA": 0,
	"EnHA": 1,
	"LAMd": 0,
	"HAMd": 0,
	"LoA": "0.0",
	"HiA": "0.0",
	“Dev”: “10.5”
	“PIvt”: 1500,
	“Val”: “1.0”,
	"Tag": "W4220_CC0001-Lt"
	}
	]
	}
	
	■ Request: **GET /sensor_config/slot_0/ch_0**
	
	Content-type: application/json
	Response: 200 OK
	{
	"Ch":0,
	"En":1,
	"Rng":4096,
	"EnA": 1,
	"AMd": 0,
	"LoA": "2.0",
	"HiA": "6.3",
	“Dev”: “10.5”
	“PIvt”: 1500,
	“Val”: “1.0”,
	"Tag": "W4220_CC0001-Temp"
	}

PUT	Single/Multi Channel Request: PUT /sensor_config/slot_index Single Channel Request: PUT /sensor_config/slot_index/ch_num [Example]: ?Request: PUT /sensor_config/slot_0 Content-type: application/json { " SCfg": [{ "Ch":0, "En":1, "Rng":4096, "EnLA": 1, "EnHA": 1, "LAMd": 0, "HAMd": 0, "LoA": "2.0", "HiA": "6.3", “Dev”: “10.5” “PIvt”: 1500, “Val”: “1.0”, "Tag": "W4220_CC0001-Temp" }, { "Ch":1, "En":1, "Rng":4128, "EnLA": 0, "EnHA": 0, "LAMd": 0, "HAMd": 0, "LoA": "0.0", "HiA": "0.0", “Dev”: “10.5” “PIvt”: 1500, “Val”: “1.0”, "Tag": "W4220_CC0001-RH" }, { "Ch":2, "En":1, "Rng":4160, "EnLA": 0, "EnHA": 1, "LAMd": 0, "HAMd": 0, "LoA": "0.0", "HiA": "0.0", “Dev”: “10.5” “PIvt”: 1500, “Val”: “1.0”, "Tag": "W4220_CC0001-Press" },

PUT (Cont.)	{
	"Ch":3,
	"En":1,
	"Rng":4192,
	"EnLA": 0,
	"EnHA": 0,
	"LAMd": 0,
	"HAMd": 0,
	"LoA": "0.0",
	"HiA": "0.0",
	“Dev”: “10.5”
	“PIvt”: 1500,
	“Val”: “1.0”,
	"Tag": "W4220_CC0001-Lt"
	}
	]
	}
	
	Response: 200 OK
	
	■ Request: **PUT /sensor_config/slot_0/ch_3**
	Content-type: application/json
	{
	"Ch":3,
	"En":1,
	"Rng":4192,
	"EnLA": 0,
	"EnHA": 0,
	"LAMd": 0,
	"HAMd": 0,
	"LoA": "0.0",
	"HiA": "0.0",
	“Dev”: “10.5”
	“PIvt”: 1500,
	“Val”: “1.0”,
	"Tag": "W4220_CC0001-Lt"
	}
	
	Response: 200 OK

PATCH

Single/Multi Channel Request:
PATCH /sensor_config/slot_index
Single Channel Request:
PATCH /sensor_config/slot_index/ch_num

[Example]:

■ Request: **PATCH /sensor_config/slot_0**

```
Content-type: application/json
{
"SCfg": [
   {
    "Ch":0,
    "LoA": "3.0"
   },
   {
    "Ch":2,
    "LoA": "2.0",
    "HiA": "8.0"
   }
 ]
}
```

Response: 200 OK

■ Request: **PATCH /sensor_config/slot_0/ch_3**

```
Content-type: application/json
{
    "Ch":3,
    "Tag": "Sensor 3"
}
```

Response: 200 OK

■ JSON array name definition:

Field	Abbreviation	Data Type
Array of Analog input configurations	SCfg	Array

■ Resource value definitions by Each Channel (Total channels = AI channel number):

Field	Abbreviation	Data Type	Property	Description
Channel Number	Ch	Number	R	0, 1, …: Sensor input channel number.
Channel Enable	En	Number	RW	1 / 0: Enable / Disable function of this sensor channel.
Input Range	Rng	Number	RW	Sensor input range. (Refer to Sensor type code table)
Enable Low Alarm	EnLA	Number	RW	1 / 0: Enable / Disable AI low alarm function
Enable High Alarm	EnHA	Number	RW	1 / 0: Enable / Disable AI high alarm function
Alarm Low Mode	LAMd	Number	RW	The alarm mode 0 Momentary 1 Latch
Alarm High Mode	HAMd	Number	RW	The alarm mode 0 Momentary 1 Latch

Low Alarm Value	LoA	String	RW	Set/get the low alarm limit value. Data format is “±xxxxxxx.yyy” Value range: -99999.999 ~ +99999.999 For example, “+0003.250” or “3.25”
High Alarm Value	HiA	String	RW	Set/get the high alarm limit value. Data format is “±xxxxxxx.yyy” Value range: -99999.999 ~ +99999.999 For example, “+0015.250” or “15.25”
Sensor Deviation Value	Dev	String	RW	Sensor deviation value Data format is “+xxxxxxx.yyy” If deviation value is 0, the module will not check deviation. (Positive only) Value range: 0 ~ 99999.999
One Measurement Interval	PItv	Number	RW	The interval of measurement. Value range: 100 ~ 8640000 (Unit:10ms) Default: 1500 *The measurement interval is retrieved the minimum value between temperature and humidity setting.
Sensor Offset Value	Val	String	RW	The sensor offset value Data format is “±xxxxxxx.yyy” Value range: -1000.000 ~ +1000.000 For example, “+0015.250” or “15.25”
Input Range for sensor offset value	RCD	Number	RW	Sensor input range for user input offset only. (Refer to Sensor type code table)
Tag Name	Tag	String	RW	The description tag for this channel. Max. 21 characters
Remarks				

B.2.10 Sensor Data Acquisition

/sensor _value/slot_0/ch_num

Table B.10: Sensor Data Acquisition	
Description	Retrieves information about the sensor input value resource on specific slot.
URL Structure	**http://10.0.0.1/sensor_value/slot_macID** **http://10.0.0.1/sensor_value/slot_macID/ch_num** **where macID = xxyyzziijjkk: the MAC ID of WISE-4210 end device (node).** **= 0: for other models** **where num = 0 ~ : the channel number**
HTTP Method	GET: Returns the representation of all of sensor input value resource. PUT: None PATCH: Apply partial modifications to sensor input value resource.

GET	Multi Channel Request:
	GET /sensor_value/slot_index
	Single Channel Request:
	GET /sensor_value/slot_index/ch_num
	
	[Example]:
	■ Request: **GET /sensor_value/slot_0**
	
	Content-type: application/json
	Response: 200 OK
	{
	"SVal": [
	{
	"Ch":0,
	"En":1,
	"Rng":4096,
	"Evt":0,
	"ClrH": 0,
	"ClrL": 0,
	"EgF":0.650,
	"HEgF":1.250,
	"LEgF":0.500,
	“LoA”: 0,
	“HiA”: 0
	},
	{
	"Ch":1,
	"En":1,
	"Rng":4128,
	"Evt":0,
	"ClrH": 0,
	"ClrL": 0
	"EgF":0.000,
	"HEgF":0.000,
	"LEgF":0.000,
	“LoA”: 0,
	“HiA”: 0
	}
	]
	}

GET (Cont.)	■ Request: **GET /sensor_value/slot_0/ch_1**
	
	Content-type: application/json
	Response: 200 OK
	{
	"Ch":1,
	"En":1,
	"Rng":4128,
	"Evt":0,
	"ClrH": 0,
	"ClrL": 0
	"EgF":0.000,
	"HEgF":0.000,
	"LEgF":0.000,
	“LoA”: 0,
	“HiA”: 0
	}

PUT	None

PATCH	Single/Multi Channel Request:
	PATCH /sensor_value/slot_index
	Single Channel Request:
	PATCH /sensor_value/slot_index/ch_num
	
	[Example]:
	■ Request: **PATCH /sensor_value/slot_0**
	
	Content-type: application/json
	{
	" SVal": [
	{
	"Ch":0,
	"ClrH": 0
	},
	{
	"Ch":1,
	"ClrL":0
	}
	]
	}
	
	Response: 200 OK
	
	■ Request: **PATCH /sensor_value/slot_0/ch_1**
	Content-type: application/json
	{
	"ClrL":0
	}
	
	Response: 200 OK

■ JSON array name definition:

Field	Abbreviation	Data Type
Array of Sensor value	SVal	Array

■ Resource value definitions (Total channels = AI channel number):

Field	Abbreviation	Data Type	Property	Description
Channel Number	Ch	Number	R	0, 1, …: Sensor input channel number.

Channel Enable	En	Number	R	1 / 0: Enable / Disable function of this sensor channel.
Input Range	Rng	Number	R	Sensor input range. (Refer to Sensor type code table)
Channel Event Status	Evt	Number	R	AI statuses Bit Order: Description 0: Fail to provide AI (sensor) value (UART timeout, ADC error) 1~31: Reserved
Clear Maximum Sensor Value	ClrH	Number	W	1: Clear the Maximum Sensor value
Clear Minimum Sensor Value	ClrL	Number	W	1: Clear the Minimum Sensor value
Channel Engineering data (floating type)	EgF	Number	R	Sensor engineering data, the value is floating type. Unit: According sensor type code For example, 999.120 -> 999.12 ºC -3.220 -> -3.22 ºC
Maximum Sensor Engineering data (floating type)	HEgF	Number	R	Sensor max. engineering data, the value is is floating type. Unit: According sensor type code For example, 10.200 -> 10.2 ºC -5.120-> -5.12 ºC
Minimum Sensor Engineering data (floating type)	LEgF	Number	R	Sensor min. engineering data, the value is floating type. Unit: According sensor type code For example, 250.350 -> 250.35 ºC -0.120 -> -0.12 ºC
Low Alarm Status	LoA	Number	RW	Low alarm status Read 1 : low alarm occurred. 0 : not occurred Write 0 : clear the low alarm status
High Alarm Status	HiA	Number	RW	High alarm status Read 1 : high alarm occurred. 0 : not occurred Write 0 : clear the high alarm status
Remarks				

B.2.11 Counter Configuration

/cnt_genconfig/slot_index

Table B.11: Counter Configuration

Description	Retrieves information about the counter configuration resource on specific slot.
URL Structure	**http://10.0.0.1/cnt_genconfig/slot_index** **where index = 0 : the core module** **1 ~ : the identifier of I/O extension slot**
HTTP Method	GET: Return the representation of all of Counter configurations resource. PUT: Replace all of Counter configurations resource. PATCH: Apply partial modifications to Counter configurations resource.

GET	Multi Channel Request:
	GET /cnt_genconfig/slot_index
	
	[Example]:
	■ Request: **GET /cnt_genconfig/slot_1**
	
	Content-type: application/json
	Response: 200 OK
	{
	"Ft": 10,
	"Fac": 1
	}

PUT	Single/Multi Channel Request:
	PUT /cnt_genconfig /slot_index
	
	[Example]:
	■ Request: **PUT /cnt_genconfig /slot_0**
	Content-type: application/json
	{
	"Ft": 30
	"Fac": 100
	}
	
	Response: 200 OK

PATCH	Single/Multi Channel Request:
	PATCH /cnt_genconfig /slot_index
	
	[Example]:
	■ Request: **PATCH /cnt_genconfig /slot_0**
	Content-type: application/json
	{
	" Ft":50
	}
	
	Response: 200 OK

■ Resource value definitions by Each Channel:

Field	Abbreviation	Data Type	Property	Description
Digital filter	Ft	Number	RW	Digital filter(0~40000 us)
Freq Acq Time	Fac	Number	RW	Frequency Acquire Timer(0~10000 ms)

Remarks

/cnt_config/slot_index/ch_num

Table B.12:	
Description	Retrieves information about the counter configuration resource on specific slot.
URL Structure	**http://10.0.0.1/cnt_config/slot_index** **http://10.0.0.1/cnt_config/slot_index/ch_num** **where index = 0 : the core module** **1 ~ : the identifier of I/O extension slot** **where num = 0 ~ : the channel number**
HTTP Method	GET: Return the representation of all of Counter configurations resource. PUT: Replace all of Counter configurations resource. PATCH: Apply partial modifications to Counter configurations resource.
GET	Multi Channel Request: **GET /cnt_config/slot_index** Single Channel Request: **GET /cnt_config/slot_index/ch_num** [Example]: ■ Request: **GET /cnt_config/slot_1** Content-type: application/json Response: 200 OK { "CntCfg": [{ "Ch":0, "Md":5, "CntIV": 16, "Act": 1, "Type": 0, "GtEn": 0, "MGt": 0, "GtTyp": 0, "TgMd": 0 }, { "Ch":1, "Md":5, "CntIV": 16, "Act": 1, "Type": 0, "GtEn": 0, "MGt": 0, "GtTyp": 0, "TgMd": 0 }, { "Ch":2, "Md":5, "CntIV": 16, "Act": 1, "Type": 0, "GtEn": 0, "MGt": 0, "GtTyp": 0, "TgMd": 0 },

	{
	"Ch":3,
	"Md":5,
	"CntIV": 16,
	"Act": 1,
	"Type": 0,
	"GtEn": 0,
	"MGt": 0,
	"GtTyp": 0,
	"TgMd": 0
	}
	]
	}

GET (Cont.)	■ Request: **GET /cnt_config/slot_0/ch_0**
	
	Content-type: application/json
	Response: 200 OK
	{
	"Ch":0,
	"Md":5,
	"CntIV": 16,
	"Act": 1,
	"Type": 0,
	"GtEn": 0,
	"MGt": 0,
	"GtTyp": 0,
	"TgMd": 0
	}

PUT	Single/Multi Channel Request: **PUT /cnt_config/slot_index** Single Channel Request: **PUT /cnt_config/slot_index/ch_num** [Example]: ■ Request: **PUT /cnt_config/slot_0** Content-type: application/json { "CntCfg": [{ "Ch":0, "Md":5, "CntIV": 16, "Act": 1, "Type": 0, "GtEn": 0, "MGt": 0, "GtTyp": 0, "TgMd": 0 },

PUT (Cont.)	{
	"Ch":1,
	"Md":5,
	"CntIV": 16,
	"Act": 1,
	"Type": 0,
	"GtEn": 0,
	"MGt": 0,
	"GtTyp": 0,
	"TgMd": 0
	},
	{
	"Ch":2,
	"Md":5,
	"CntIV": 16,
	"Act": 1,
	"Type": 0,
	"GtEn": 0,
	"MGt": 0,
	"GtTyp": 0,
	"TgMd": 0
	},
	{
	"Ch":3,
	"Md":5,
	"CntIV": 16,
	"Act": 1,
	"Type": 0,
	"GtEn": 0,
	"MGt": 0,
	"GtTyp": 0,
	"TgMd": 0
	}
	]
	}
	
	Response: 200 OK
	
	■ Request: **PUT /cnt_config/slot_0/ch_3**
	Content-type: application/json
	{
	"Ch":3,
	"Md":5,
	"CntIV": 16,
	"Act": 1,
	"Type": 0,
	"GtEn": 0,
	"MGt": 0,
	"GtTyp": 0,
	"TgMd": 0
	}
	
	Response: 200 OK

PATCH

Single/Multi Channel Request:
PATCH /cnt_config/slot_index
Single Channel Request:
PATCH /cnt_config/slot_index/ch_num

[Example]:

■ Request: **PATCH /cnt_config/slot_0**

```
Content-type: application/json
{
"CntCfg": [
   {
     "Ch":0,
     "Md":5
   },
   {
     "Ch":2,
     "CntIV": 32,
     "Act": 1
   }
 ]
}
```

Response: 200 OK

■ Request: **PATCH /cnt_config/slot_0/ch_3**

```
Content-type: application/json
{
     "Ch":3,
     "Md":5,
     "CntIV": 16,
     "Act": 0,
     "Type": 0
}
```

Response: 200 OK

■ JSON array name definition:

Field	Abbreviation	Data Type
Array of Counter configurations	CntCfg	Array

■ Resource value definitions by Each Channel (Total channels = AI channel number + 1 average channel):

Field	Abbreviation	Data Type	Property	Description
Channel Number	Ch	Number	R	0, 1, …: Counter channel number.
Mode	Md	Number	RW	Counter Mode

Value	Mode	Note
0	Bi direction	
1	Up and down	
2	Up	
3	High frequency	
4	A/B-1X	
5	A/B-2X	
6	A/B-4X	
7	Low frequency	
8	Wave width	

Startup Value	CntIV	Number	R/W	Counter Startup Value
Counter Action	Act	Number	RW	Counter Action

Value	Mode	Note
0	Stop	
1	Start	
2	Reset	
3	Clear Flag	

Counter Type	Type	Number	RW	1 or 0 of each bit: Enable or Disable the Counter Type

Bit Order	Description
0	0: Once, 1:Repeat
1	0: Reload to Zero, 1: Reload To Startup

Gate Enable	GtEn	Number	R/W	1 / 0: Enable/Disable Counter Gate Setting
Mapping gate	MGt	Number	R/W	The gate that counter is mapped to
Gate active Type	GtTyp	Number	RW	Counter Gate Active Type

Value	Type	Note
0	Low level	
1	Falling edge	
2	High level	
3	Rising edge	

Trigger Mode	TgMd	Number	RW	Counter Gate Active Type

Value	Type	Note
0	Non re-trigger	
1	Re-trigger	
2	Edge start	

Channel Number	Ch	Number	R	0, 1, …: Counter channel number.
Mode	Md	Number	RW	Counter Mode

Value	Mode	Note
0	Bi direction	
1	Up and down	
2	Up	
3	High frequency	
4	A/B-1X	
5	A/B-2X	
6	A/B-4X	
7	Low frequency	
8	Wave width	

Startup Value	CntIV	Number	R/W	Counter Startup Value
Counter Action	Act	Number	R/W	Counter Action

Value	Mode	Note
0	Stop	
1	Start	
2	Reset	

				3 | Clear Flag |
Counter Type	Type	Number	R/W	1 or 0 of each bit: Enable or Disable the Counter Type Bit Order | Description 0 | 0: Once, 1:Repeat 1 | 0: Reload to Zero, 1: Reload To Startup
Gate Enable	GtEn	Number	R/W	1 / 0: Enable/Disable Counter Gate Setting
Mapping gate	MGt	Number	R/W	The gate that counter is mapped to
Gate active Type	GtTyp	Number	R/W	Counter Gate Active Type Value | Type | Note 0 | Low level | 1 | Falling edge | 2 | High level | 3 | Rising edge |
Trigger Mode	TgMd	Number	R/W	Counter Gate Active Type Value | Type | Note 0 | Non re-trigger | 1 | Re-trigger | 2 | Edge start |
Remarks				

B.2.12 Counter Data Acquisition

/cnt_value/slot_index/ch_num

Table B.13: Counter Data Acquisition	
Description	Retrieves information about the Counter value resource on specific slot.
URL Structure	**http://10.0.0.1/cnt_value/slot_index** **http://10.0.0.1/cnt_value/slot_index/ch_num** **where index = 0 : the core module** **1 ~ : the identifier of I/O extension slot** **where num = 0 ~ : the channel number**
HTTP Method	GET: Returns the representation of all of Counter value resource. PUT: None PATCH: Apply partial modifications to Counter value resource.

GET	Multi Channel Request:
	GET /cnt_value/slot_index
	Single Channel Request:
	GET /cnt_value/slot_index/ch_num
	
	[Example]:
	■ Request: **GET /cnt_value/slot_0**
	
	Content-type: application/json
	Response: 200 OK
	{
	"CntVal": [
	{
	"Ch":0,
	"Stat": 1,
	"Val":0
	},
	{
	"Ch":1,
	"Stat": 1,
	"Val":0
	},
	{
	"Ch":2,
	"Stat": 1,
	"Val":0
	},
	{
	"Ch":3,
	"Stat": 1,
	"Val":0
	}
	]
	}
	
	■ Request: **GET /cnt_value/slot_0/ch_2**
	
	Content-type: application/json
	Response: 200 OK
	{
	"Ch":2,
	"Stat": 1,
	"Val":0
	}

<table>
<tr><td>PUT</td><td>None</td></tr>
<tr><td>PATCH</td><td>None</td></tr>
<tr><td colspan="2">■ JSON array name definition:
<table>
<tr><th>Field</th><th>Abbreviation</th><th>Data Type</th></tr>
<tr><td>Array of Counter configurations</td><td>CntVal</td><td>Array</td></tr>
</table>
■ Resource value definitions (Total channels = AI channel number + 1 average channel):
<table>
<tr><th>Field</th><th>Abbreviation</th><th>Data Type</th><th>Property</th><th>Description</th></tr>
<tr><td>Channel Number</td><td>Ch</td><td>Number</td><td>R</td><td>0, 1, …: Counter channel number.</td></tr>
<tr><td>Counter Status</td><td>Stat</td><td>Number</td><td>R</td><td>Counter Status
<table>
<tr><th>Value</th><th>Mode</th><th>Note</th></tr>
<tr><td>0</td><td>None</td><td></td></tr>
<tr><td>1</td><td>Normal</td><td></td></tr>
<tr><td>8</td><td>Over flow</td><td></td></tr>
<tr><td>9</td><td>Under flow</td><td></td></tr>
<tr><td>10</td><td>Over and Under flow</td><td></td></tr>
</table></td></tr>
<tr><td>Channel Raw Value</td><td>Val</td><td>Number</td><td>R</td><td>0 ~ 65535 :Counter measurement data (Raw data)</td></tr>
</table></td></tr>
<tr><td>Remarks</td><td></td></tr>
</table>

B.2.13 RS-485 Configuration

/serial_config/com_x

Table B.14: RS-485 Configuration	
Description	Retrieves the configuration for RS-485 format
URL Structure	**http://10.0.0.1/serial_config/com_x** **where x = 1 ~ : the identifier of COM number**
HTTP Method	GET: Returns the representation of all of RS-485 configuration resource. PUT: Replace all of RS-485 configuration resource. PATCH: Apply partial modifications to RS-485 configuration resource.

GET	Request:
	GET /serial_config/com_x
	
	[Example]:
	■ Request: **GET /serial_config/com_1**
	
	Content-type: application/json
	Response: 200 OK
	{
	“BR”:2,
	“DB”:1,
	“P”: 0,
	“SB”:0,
	“Prot”:0
	}

PUT	Request:
	PUT /serial_config/com_x
	
	[Example]:
	■ Request: **PUT /serial_config/com_1**
	Content-type: application/json
	{
	“BR”:2,
	“DB”:1,
	“P”: 0,
	“SB”:0,
	“Prot”:0
	}
	
	Response: 200 OK

PATCH	Request: **PATCH /serial_config/com_x**
	
	[Example]:
	■ Request: **PATCH /serial_config/com_1**
	Content-type: application/json
	{
	“BR”:7,
	}
	
	Response: 200 OK

■ Resource value definitions:

Field	Abbreviation	Data Type	Property	Description
Baud rate	BR	Number	RW	Baud rate setting for RS-485 Default: 3->9600 bps

0	1200 bps
1	2400 bps
2	4800 bps
3	9600 bps
4	19200 bps
5	38400 bps
6	57600 bps
7	115200 bps

Field	Abbreviation	Data Type	Property	Description
Data bit	DB	Number	RW	Data bit for RS-485 Default: 1->8 bit

0	7 bit
1	8 bit

Field	Abbreviation	Data Type	Property	Description
Parity	P	Number	RW	Parity for RS-485 Default: 0->None

0	None
1	Odd
2	Even

Field	Abbreviation	Data Type	Property	Description
Stop bit	SB	Number	RW	Stop bit for RS-485 Default: 0->1 bit

0	1 Stop bit
1	2 Stop bit

<table>
<tr><td>Protocol</td><td>Prot</td><td>Number</td><td>RW</td><td>The protocol used for RS-485<table><tr><td>0</td><td>Modbus/RTU (Client)</td></tr></table></td></tr>
<tr><td>Support</td><td>FCS</td><td>Number</td><td>RW</td><td>Quick configuration or function support of RS-485 COM port.
Default: 0->None<table><tr><td>Bit</td><td>R/W</td><td>Description</td></tr><tr><td>0</td><td>W</td><td>Quick config. setup for WISE-2460P (Write Only)</td></tr><tr><td></td><td>R</td><td>1: Device WISE-2460P is detected.
0: WISE-2460P No found
(Read Only)</td></tr><tr><td>1</td><td>W</td><td>Quick config. setup for WISE-S100 (Write Only)</td></tr><tr><td></td><td>R</td><td>1: Device WISE-S100 is detected.
0: WISE-S100 No found
(Read Only)</td></tr><tr><td>2</td><td>W</td><td>Quick config. setup for WISE-2460 (Write Only)</td></tr><tr><td></td><td>R</td><td>1: Device WISE-2460 is detected.
0: WISE-2460 No found
(Read Only)</td></tr><tr><td>3 ~</td><td>-</td><td>Reserved</td></tr></table></td></tr>
<tr><td colspan="5">Remarks</td></tr>
</table>

B.2.14 Modbus/RTC General Configuration

/modbusserver_genconfig/com_x

Table B.15: Modbus/RTC General Configuration	
Description	Retrieves the configuration for Modbus/RTU general configuration
URL Structure	**http://10.0.0.1/modbusserver_genconfig/com_x** **where x = 1 ~ : the identifier of COM number**
HTTP Method	GET: Returns the representation of all of RS-485 general configuration resource. PUT: Replace all of RS-485 general configuration resource. PATCH: Apply partial modifications to RS-485 general configuration resource.

<table>
<tr><td>GET</td><td>Request:
GET /modbusserver_genconfig/com_x

[Example]:
■ Request: GET /modbusserver_genconfig/com_1

Content-type: application/json
Response: 200 OK
{
“RT”: 100,
“DBP”: 50,
“EnC”:1
}</td></tr>
<tr><td>PUT</td><td>Request:
PUT /modbusserver_genconfig/com_x

[Example]:
■ Request: PUT /modbusserver_genconfig/com_1
Content-type: application/json
{
“RT”: 100,
“DBP”: 50,
“EnC”:1
}

Response: 200 OK</td></tr>
<tr><td>PATCH</td><td>Request: PATCH /modbusserver_genconfig/com_x

[Example]:
■ Request: PATCH /modbusserver_genconfig/com_1
Content-type: application/json
{
“BR”:7,
}

Response: 200 OK</td></tr>
<tr><td colspan="2">■ Resource value definitions:
<table>
<tr><th>Field</th><th>Abbreviation</th><th>Data Type</th><th>Property</th><th>Description</th></tr>
<tr><td>Server response timeout</td><td>RT</td><td>Number</td><td>RW</td><td>The maximum time for waiting server device response
Receive timeout: 50-5000 ms
Default: 200 ms</td></tr>
<tr><td>Delay Between Polls</td><td>DBP</td><td>Number</td><td>RW</td><td>Delay time for sending next request
Delay time: 1-1000 ms
Default: 200 ms</td></tr>
<tr><td>Enable Modbus RTU CRC check</td><td>EnC</td><td>Number</td><td>RW</td><td>1 / 0: Enable / Disable CRC checksum verification</td></tr>
</table></td></tr>
<tr><td>Remarks</td><td></td></tr>
</table>

B.2.15 Modbus/RTU Polling Rules Setting

/modbusserver_config/slot_0/com_x

/modbusserver_config/slot_0/com_x/idx_num

Table B.16: Modbus/RTU Polling Rules Setting	
Description	Retrieves information about the Modbus/RTU coil poll setting
URL Structure	**http://10.0.0.1/modbusserver_config/slot_0/com_x** **http://10.0.0.1/modbusserver_config/slot_0/com_x/seg_y http://10.0.0.1/modbus-server_config/slot_0/com_x/idx_num** **where x = 1 ~ : the identifier of COM number** **where y= 0 ~ : the index of data resource requisition, 8 rules in each segment** **where num = 0 ~ 20 : the rule number**
HTTP Method	GET: Returns the representation of all of Modbus/RTU coil poll resource. PUT: Replace all of all of Modbus/RTU coil poll resource PATCH: Apply partial modifications to Modbus/RTU coil poll resource.
GET	Multi-Channel Request: **GET /modbusserver_config/slot_0/com_x** **GET /modbusserver_config/com_x (only get first 8 rules)** **GET /modbusserver_config/com_x/seg_0 (get rule 0~ 7)** **GET/modbusserver_config/com_x/seg_1 (get rule 8 ~ 15)** **GET/modbusserver_config/com_x/seg_2 (get rule 16 ~ 19)** Single Channel Request: **GET /modbusserver_config/slot_0/com_x/idx_num** [Example]: Request: **GET /modbusserver_config/slot_0/com_1** Request: **GET /modbusserver_config/slot_0/com_1/seg_0** Content-type: application/json Response: 200 OK { "RtuCfg": [{ “Idx”:0 “SID”:1, “FC”:1, “Addr”:1, “NOP”:4, “SItv”:1000, “Prop”:1, “MCh”:0, “LgE”: 0, “DevE”:0, "Dev":300 },

GET (Cont.)	{ “Idx”:1 “SID”:2, “FC”:1, “Addr”:1, “NOP”:4, “SItv”:1000, “Prop”:1, “MCh”:4, “LgE”: 0, “DevE”:0, "Dev":3276 }, { “Idx”:2 “SID”:3, “FC”:1, “Addr”:17, “NOP”:4, “SItv”:1000, “Prop”:1, “MCh”:8, “LgE”: 0, “DevE”:0, "Dev":100 }, { "Idx":3, “SID”:3, “FC”:1, “Addr”:2, “NOP”:4, “SItv”:1000, “Prop”:1, “MCh”:12, “LgE”: 0, “DevE”:0, "Dev":30 }, { "Idx":4, “SID”:0, “FC”:0, “Addr”:0, “NOP”:1, “SItv”:1000, “Prop”:1, “MCh”:255, “LgE”: 0, “DevE”:0, "Dev":3276 }, ……

	{
	"Idx":7,
	“SID”: 0,
	“FC”:0,
	“Addr”:0,
	“NOP”:1,
	“SItv”:1000,
	“Prop”:1,
	“MCh”:255,
	“LgE”: 0,
	“DevE”:0,
	"Dev":300
	}
	]
	}

GET (Cont.)	■ Request: **GET /modbusserver_config/slot_0/com_1/idx_2**
	
	Content-type: application/json
	Response: 200 OK
	{
	"Idx":2,
	“SID”:3,
	“FC”:1,
	“Addr”:1,
	“NOP”:4,
	“SItv”:1000,
	“Prop”:1
	“MCh”:8,
	“LgE”: 0,
	“DevE”:0,
	"Dev":300
	}

PUT	Multi-Channel Request: **PUT /modbusserver_config/slot_0/com_x** **PUT /modbusserver_config/slot_0/com_x/seg_y** Single Channel Request: **PUT /modbusserver_config/slot_0/com_x/idx_num** [Example]: ■ Request: **PUT /modbusserver_config/slot_0/com_1** ■ Request: **PUT /modbusserver_config/slot_0/com_1/seg_0** Content-type: application/json { "RtuCfg": [{ “Idx”:0 “SID”:1, “FC”:1, “Addr”:1, “NOP”:4, “SItv”:1000, “Prop”:1, “MCh”:0, “LgE”: 0, “DevE”:0, "Dev":3276 }, { “Idx”:1 “SID”:2, “FC”:1, “Addr”:1, “NOP”:4, “SItv”:1000, “Prop”:1, “MCh”:4, “LgE”: 0, “DevE”:0, "Dev":500 }, { “Idx”:2 “SID”:3, “FC”:1, “Addr”:1, “NOP”:4, “SItv”:1000, “Prop”:1, “MCh”:8, “LgE”: 0, “DevE”:0, "Dev":1 },

PUT (Cont.)	{
	"Idx":3,
	“SID”:3,
	“FC”:1,
	“Addr”:2,
	“NOP”:4,
	“SItv”:1000,
	“Prop”:1,
	“MCh”:12,
	“LgE”: 0,
	“DevE”:0,
	"Dev":300
	},
	{
	"Idx":4,
	“SID”:0,
	“FC”:0,
	“Addr”:0,
	“NOP”:1,
	“SItv”:1000,
	“Prop”:1,
	“MCh”:255,
	“LgE”: 0,
	“DevE”:0,
	"Dev":3276
	},
	……
	{
	"Idx":7,
	“SID”: 0,
	“FC”:0,
	“Addr”:0,
	“NOP”:1,
	“SItv”:1000,
	“Prop”:1,
	“MCh”:255,
	“LgE”: 0,
	“DevE”:0,
	"Dev":300
	}
	]
	}
	
	Response: 200 OK

PUT (Cont.)	■ Request: **PUT /modbusserver_config/slot_0/com_1/idx_2**
	Content-type: application/json
	{
	"Idx":3,
	“SID”:3,
	“FC”:1,
	“Addr”:2,
	“NOP”:4,
	“SItv”:1000,
	“Prop”:1,
	“MCh”:12,
	“LgE”: 0,
	“DevE”:0,
	"Dev":300
	}
	
	Response: 200 OK

PATCH	Single/Multi Channel Request:
	PATCH /modbusserver_config/slot_0/com_x
	Single Channel Request:
	PATCH /modbusserver_config/slot_0/com_x/idx_num
	
	[Example]:
	■ Request: **PATCH /modbusserver_config/slot_0/com_1**
	Content-type: application/json
	{
	"RtuCfg": [
	{
	"Idx":2,
	“SID”:1
	},
	{
	"Idx":3,
	“SItv”:500
	}
	]
	}
	
	Response: 200 OK
	
	?Request: PATCH /modbusserver_config/slot_0/com_1/idx_2
	Content-type: application/json
	{
	“Addr”:211
	}
	
	Response: 200 OK

■ JSON array name definition:

Field	Abbreviation	Data Type
Array of Modbus RTU poll configurations	RtuCfg	Array

■ Resource value definitions (Total index = 8):

Field	Abbreviation	Data Type	Property	Description
Item Index	Idx	Number	R	0, 1, …7: Rule Index number of Modbus RTU set

<table>
<tr><td>Server ID</td><td>SID</td><td>Number</td><td>RW</td><td>Modus RTU Server device ID
1-255</td></tr>
<tr><td>Function code</td><td>FC</td><td>Number</td><td>RW</td><td>Modbus RTU function code used for polling
Default: 0->Disable<table><tr><td>0</td><td>Disable</td></tr><tr><td>1</td><td>Coil status</td></tr><tr><td>2</td><td>Input status</td></tr><tr><td>3</td><td>Holding register</td></tr><tr><td>4</td><td>Input register</td></tr></table></td></tr>
<tr><td>Address</td><td>Addr</td><td>Number</td><td>RW</td><td>Modbus RTU polling address
1-65535
Default: 1</td></tr>
<tr><td>Number of points</td><td>NOP</td><td>Number</td><td>RW</td><td>The number of the address length
1-32
Default: 1</td></tr>
<tr><td>Scan interval</td><td>SItv</td><td>Number</td><td>RW</td><td>1~65535 s
Default: 60 s</td></tr>
<tr><td>R/W property</td><td>Prop</td><td>Number</td><td>RW</td><td>RW property of the polling address<table><tr><td>0</td><td>Read only</td></tr><tr><td>1</td><td>Write only</td></tr><tr><td>2</td><td>Read or write</td></tr></table></td></tr>
<tr><td>Mapping channel</td><td>MCh</td><td>Number</td><td>RW</td><td>The mapping start channel number for RESTful request
0~31
Default: 0</td></tr>
<tr><td>Log enable</td><td>LgE</td><td>Number</td><td>RW</td><td>0/1 : Enable/disable Modbus RTU log function</td></tr>
<tr><td>Deviation and COS enable</td><td>DevE</td><td>Number</td><td>RW</td><td>0/1:
Enable/disable Modbus RTU “deviation” for registers, and “change of state” for coils.</td></tr>
<tr><td>Raw Data Deviation Value</td><td>Dev</td><td>Number</td><td>RW</td><td>The deviation value of register raw data.
1 ~ 65535
Default: 3276</td></tr>
<tr><td colspan="5">Remarks</td></tr>
</table>

B.2.16 Modbus/RTU Response Status

/modbusserver_status/com_x

/modbusserver_status/com_x/idx_num

Table B.17: Modbus/RTU Response Status	
Description	Retrieves information about the Modbus/RTU coil poll status
URL Structure	**http://10.0.0.1/modbusserver_status/com_x** **http://10.0.0.1/ modbusserver_status/com_x/idx_num** **where x = 1 ~ : the identifier of COM number** **where num = 0 ~ : the channel number**
HTTP Method	GET: Returns the representation of all of Modbus/RTU polling status resource. PUT: None PATCH: None
GET	Multi-Channel Request: **GET / modbusserver_status/com_x** Single Channel Request: **GET / modbusserver_status/com_x/idx_num** [Example]: Request: **GET / modbusserver_status/com_1** Content-type: application/json Response: 200 OK { "RtuStat": [{ “Idx”:0, “CRT”:33, “HRT”:35, “LRT”:32, “Evt”:0 }, { “Idx”:1, “CRT”:9, “HRT”:11, “LRT”:6, “Evt”:0 }, { “Idx”:2, “CRT”:9, “HRT”:11, “LRT”:6, “Evt”:0 }, { “Idx”:3, “CRT”:9, “HRT”:11, “LRT”:6, “Evt”:0 }, ……

<table>
<tr><td>GET</td><td>{
"Idx":7,
“CRT”:0,
“HRT”:0,
“LRT”:65535,
“Evt”:16
}
]
}

■ Request: GET /modbusserver_status/com_1/idx_2
Content-type: application/json
Response: 200 OK
{
“Idx”:2,
“CRT”:9,
“HRT”:11,
“LRT”:6,
“Evt”:0
}</td></tr>
<tr><td>PUT</td><td>None</td></tr>
<tr><td>PATCH</td><td>None</td></tr>
<tr><td colspan="2">■ JSON array name definition:
<table><tr><th>Field</th><th>Abbreviation</th><th>Data Type</th></tr><tr><td>Array of Modbus RTU polling status</td><td>RtuStat</td><td>Array</td></tr></table>
■ Resource value definitions (Total index = 8):
<table><tr><th>Field</th><th>Abbreviation</th><th>Data Type</th><th>Property</th><th>Description</th></tr><tr><td>Item Index</td><td>Idx</td><td>Number</td><td>R</td><td>0, 1, …7: Index number of Modbus RTU set</td></tr><tr><td>The last (current) server device response time</td><td>CRT</td><td>Number</td><td>R</td><td>0~65535: The last response time of polling server device.
Unit: ms</td></tr><tr><td>The max server device response time</td><td>HRT</td><td>Number</td><td>R</td><td>0~65535: The maximum response time of polling server device.
Unit: ms</td></tr><tr><td>The min server deivce response time</td><td>LRT</td><td>Number</td><td>R</td><td>0~65535: The minimum response time of polling server device.
Unit: ms</td></tr><tr><td>Modbus RTU Polling Status Code</td><td>Evt</td><td>Number</td><td>R</td><td>The status code of Modbus RTU polling</td></tr></table></td></tr>
<tr><td>Remarks</td><td></td></tr>
</table>

B.2.17 Modbus/RTU Status Code

Table B.18: Modbus/RTU Status Code				
Event Value	**Description**	**Polling Status**	**Read function**	**Write function**
0 (0x00)	No error	V	V	V
1 (0x01)	Illegal function	V	V	V
2 (0x02)	Illegal data address	V	V	V
3 (0x03)	Illegal data value	V	V	V
4 (0x04)	Server device failure	V	V	V
5 (0x05)	Acknowledge	V	V	V
6 (0x06)	Server device busy	V	V	V
7 (0x07)	Negative acknowledge	V	V	V
8 (0x08)	Memory parity error	V	V	V
9 (0x09)	Reserved			
10 (0x0A)	Gateway path unavailable	V	V	V
11 (0x0B)	Gateway target device failed to respond	V	V	V
12 ~15	Reserved			
16 (0x10)	Unavailable	V	V	
17 (0x11)	Server response timeout	V	V	V
18 (0x12)	Checksum error	V	V	V
19 (0x13)	Received data error	V	V	V
20 (0x14)	Send request fail	V	V	V
21(0x15)	Unprocessed	V	V	V
22(0x16)	Read only			V
23(0x17)	In processing			V
24(0x18)	Invalid Protocol			V

B.2.18 Expansion tag data – bit data

/expansion_bit/slot_0/com_x

/expansion_bit/slot_0/com_x/idx_y

/expansion_bit/slot_0/com_x/ch_num

Table B.19: Expansion tag data – bit data	
Description	Retrieves information about the expansion tag bit data resource, the data information is defined by user configuration
URL Structure	**http://10.0.0.1/expansion_bit/slot_0/com_x (GET only)** **http://10.0.0.1/expansion_bit/slot_0/com_x/idx_y (GET only)** **http://10.0.0.1/expansion_bit/slot_0/com_x /ch_num** **where x = 1 ~ : the identifier of COM number** **where y= 0 ~ : the index of data resource requisition, 8 channel in each index** **where num = 0 ~ : the channel number**
HTTP Method	GET: Returns the representation of all of expansion bit data resource. PUT: None. PATCH: Apply partial modifications to expansion bit data resource.
GET	Multi-Channel Request: **GET /expansion_bit/slot_0/com_x (only get first 32 channel)** **GET /expansion_bit/slot_0/com_x/idx_0 (get first 8 channel)** **GET /expansion_bit/slot_0/com_x/idx_1 (get second 8 channel)** … Single Channel Request: **GET /expansion_bit/slot_0/com_x/ch_num** [Example]: ■ Request: **GET /expansion_bit/slot_0/com_1** Content-type: application/json Response: 200 OK { "ExpBit": [{ "Ch":0, “Val”:1, “Evt”:0, “SID”:1, “Addr”:1, “MAddr”:1001, “WEvt”:0 }, { "Ch":1, “Val”:0, “Evt”:0, “SID”:1, “Addr”:2, “MAddr”:1002, “WEvt”:0 },

GET (Cont.)


```
    {
      "Ch":2,
      “Val”:1,
      “Evt”:0,
      “SID”:1,
      “Addr”:3,
      “MAddr”:1003,
      “WEvt”:0
    },
    {
      "Ch":3,
      “Val”:1,
      “Evt”:0,
      “SID”:1,
      “Addr”:4,
      “MAddr”:1004,
      “WEvt”:0
    },
    {
      "Ch":4,
      “Val”:1,
      “Evt”:0,
      “SID”:2,
      “Addr”:1,
      “MAddr”:1005,
      “WEvt”:0
    },
    ……
{
      "Ch":31,
      “Val”:0,
      “Evt”:0,
      “SID”:3,
      “Addr”:17,
      “MAddr”:1032,
      “WEvt”:0
    }
  ]
}
```


■ Request: **GET /expansion_bit/slot_0/com_1/idx_4**


```
{
  "ExpBit": [
    {
      "Ch":32,
      “Val”:1,
      “Evt”:0,
      “SID”:1,
      “Addr”:1,
      “MAddr”:1001,
      “WEvt”:0
    },
```

<table>
<tr><td></td><td>{
"Ch":33,
“Val”:0,
“Evt”:0,
“SID”:1,
“Addr”:2,
“MAddr”:1002,
“WEvt”:0
},
……
{
"Ch":39,
“Val”:0,
“Evt”:0,
“SID”:3,
“Addr”:17,
“MAddr”:1032,
“WEvt”:0
}
]
}

■ Request: GET /expansion_bit/slot_0/com_1/ch_2

Content-type: application/json
Response: 200 OK
{
"Ch":2,
“Val”:1,
“Evt”:0,
“SID”:1,
“Addr”:3,
“MAddr”:1003,
“WEvt”:0
}</td></tr>
<tr><td>PUT</td><td>None</td></tr>
<tr><td>PATCH</td><td>Single Channel Request:
PATCH /expansion_bit/slot_0/com_x/ch_num

[Example]:
■ Request: PATCH /expansion_bit/slot_0/com_1/ch_3
Content-type: application/json
{
"Val":1
}

Response: 200 OK</td></tr>
</table>

■ JSON array name definition:

Field	Abbreviation	Data Type
Array of Modbus RTU polling status	RtuStat	Array

■ Resource value definitions (Total channels = 32):

Field	Abbreviation	Data Type	Property	Description
Channel Number	Ch	Number	R	0, 1, …: expansion tag data channel number

Channel Value	Val	Number	R/W	The channel value of expansion tag data Value: 0/1 *After writing action, user must poll the “expansion bit writing status” to get process result.
Channel Status Code for Read Operation	Evt	Number	R	The channel status code of expansion tag data (Refer to Modbus/RTU Status Code)
Server ID	SID	Number	R	Modus RTU Server device ID 0-255
Server Modbus Address	Addr	Number	R	Modbus RTU Server device polling address 1-9999
Modbus TCP Mapping Address	MAddr	Number	R	Modbus TCP mapping address of expansion value 1-9999
Expansion bit writing status	WEvt	Number	R	The status for preview writing action. (Refer to Modbus/RTU Status Code) *Event value 0x17: the writing process is not finish, user should poll the status later.
Remarks				

B.2.19 Expansion tag data – word data

/expansion_word/slot_0/com_x/ch_num

Table B.20: Expansion tag data – word data	
Description	Retrieves information about the expansion word data resource, the data information is defined by user configuration
URL Structure	**http://10.0.0.1/expansion_word/slot_0/com_x (GET only)** **http://10.0.0.1/expansion_word/slot_0/com_x/idx_y (GET only)** **http://10.0.0.1/expansion_word/slot_0/com_x/ch_num** **where x = 1 ~ : the identifier of COM number** **where y= 0 ~ : the index of data resource requisition, 8 channel in each index** **where num = 0 ~ : the channel number**
HTTP Method	GET: Returns the representation of all of expansion word data resource. PUT: None PATCH: Apply partial modifications to expansion word data resource.
GET	Multi-Channel Request: **GET / expansion_word/slot_0/com_x (first 32 channel)** **GET / expansion_word /slot_0/com_x/idx_0 (get first 8 channel)** **GET / expansion_word /slot_0/com_x/idx_1 (get second 8 channel)** … Single Channel Request: **GET /expansion_word/slot_0/com_x/ch_num** [Example]: ■ Request: **GET /expansion_word/slot_0/com_1** Content-type: application/json Response: 200 OK { "ExpWord": [{ "Ch":0, "Val":32768, “Evt”:0, “SID”:1, “Addr”:1, “MAddr”:1001, “WEvt”:0 }, { "Ch":1, "Val":1235, “Evt”:0, “SID”:1, “Addr”:2, “MAddr”:1002, “WEvt”:0 },

GET (Cont.)


```
  {
    "Ch":2,
    "Val":65535,
    “Evt”:0,
    “SID”:1,
    “Addr”:3,
    “MAddr”:1003,
    “WEvt”:0
  },
  {
    "Ch":3,
    "Val":33358,
    “Evt”:0,
    “SID”:1,
    “Addr”:4,
    “MAddr”:1004,
    “WEvt”:0
  },
  {
    "Ch":4,
    "Val":4095,
    “Evt”:0,
    “SID”:2,
    “Addr”:211,
    “MAddr”:1005,
    “WEvt”:0
  },
  ……
{
    "Ch":31,
    "Val":0,
    “Evt”:0,
    “SID”:3,
    “Addr”:1,
    “MAddr”:1032,
    “WEvt”:0
  }
 ]
}
```


■ Request: **GET /expansion_word/slot_0/com_1/idx_0**


```
{
  "ExpWord": [
    {
      "Ch": 0,
      "Val": 32767,
      "Evt": 0,
      "SID": 2,
      "Addr": 1,
      "MAddr": 1001,
      "WEvt": 0
    },
    …
```

<table>
<tr><td>GET (Cont.)</td><td>
<pre>
      {
         "Ch": 7,
         "Val": 65535,
         "Evt": 0,
         "SID": 2,
         "Addr": 8,
         "MAddr": 1008,
         "WEvt": 0
      }
   ]
}
</pre>
■ Request: GET /expansion_word/slot_0/com_1/ch_2

Content-type: application/json
Response: 200 OK
<pre>
{
    "Ch":2,
    "Val":65535,
    “Evt”:0,
    “SID”:1,
    “Addr”:3,
    “MAddr”:1003
}
</pre>
</td></tr>
<tr><td>PUT</td><td>None</td></tr>
<tr><td>PATCH</td><td>Single Channel Request:
PATCH /expansion_word/slot_0/com_x/ch_num

[Example]:
■ Request: PATCH /expansion_word/slot_0/com_1/ch_3
Content-type: application/json
<pre>
{
    "Val":32768
}
</pre>
Response: 200 OK</td></tr>
</table>

■ JSON array name definition:

Field	Abbreviation	Data Type
Array of Analog input configurations	ExpWord	Array

■ Resource value definitions (Total channels = 32):

Field	Abbreviation	Data Type	Property	Description
Channel Number	Ch	Number	R	0, 1, …: expansion tag data channel number
Channel Value	Val	Number	R/W	The channel value of expansion data *After writing action, user must poll the “expansion bit writing status” to get process result.
Channel Status Code for Read Operation	Evt	Number	R	The channel status code of expansion word value (Refer to Modbus/RTU Status Code)
Server ID	SID	Number	R	Modus RTU server device ID 0-255

Server Modbus Address	Addr	Number	R	Modbus RTU server device polling address 1-9999
Modbus TCP Mapping Address	MAddr	Number	R	Modbus TCP mapping address of expansion value 1-9999
Expansion word writing status	WEvt	Number	R	The status for preview writing action. (Refer to Modbus/RTU Status Code) *Event value 0x17: the writing process is not finish, user should poll the status later.
Remarks				

B.2.20 Network

Table B.21: Network	
Description	Retrieves the network link information.
URL Structure	**http://10.0.0.1/net_basic**
HTTP Method	GET: Returns the representation of all of network link information resource. PUT: Replace all of network link information resource. PATCH: Apply partial modifications to network link information resource.

GET	Request:
	GET /net_basic
	
	[Example]:
	■ Request: GET /net_basic
	
	Content-type: application/json
	Response: 200 OK
	{
	"MAC":"00-D0-C9-F0-63-F7",
	"DHCP":0,
	"IP":"10.0.0.1",
	"Msk":"255.255.0.0",
	"GW":"0.0.0.0"
	}

PUT	Request:
	PUT /net_basic
	
	[Example]:
	■ Request: **PUT /net_basic**
	Content-type: application/json
	{
	"MAC":"00-D0-C9-F0-63-F7",
	"DHCP":0,
	"IP":"192.168.0.1",
	"Msk":"255.255.255.0",
	"GW":"0.0.0.0"
	}
	Response: 200 OK

PATCH	Request: **PATCH /net_basic**
	
	[Example]:
	■ Request: **PATCH /net_basic**
	Content-type: application/json
	{
	"DHCP":1
	}
	
	Response: 200 OK

■ Resource value definitions :

Field	Abbreviation	Data Type	Property	Description
MAC ID	MAC	String	R	MAC address. (12+5) characters, ex, "00-D0-C9-F0-63-F7"
Get IP Method	DHCP	Number	RW	Get IP Method 0: static IP 1: DHCP 2: Static IP According to Dip Switch Settings (ADAM-3600-A1F Only)
IP Address	IP	String	RW	IP v4 IP address
Subnet Mask	Msk	Msk	Msk	IP v4 Subnet Mask
Gateway Address	GW	String	RW	IP v4 Gateway address

Remarks

B.2.21 Wi-Fi Wireless network

/wlan_config

Table B.22: Wi-Fi Wireless network	
Description	Retrieves the wireless network configuration.
URL Structure	**http://10.0.0.1/wlan_config**
HTTP Method	GET: Returns the representation of all of WLAN configuration resource. PUT: Replace all of WLAN configuration resource. PATCH: Apply partial modifications to WLAN configuration resource.

GET	Request:
	GET /wlan_config
	
	[Example]:
	■ Request: **GET /wlan_config**
	
	Content-type: application/json
	Response: 200 OK
	{
	"Md":0,
	"ISSID":"corega",
	"ISec":2,
	"IKey":"12345678",
	"ISSID2":"Giant",
	"ISec2":0,
	"IKey2":"",
	"ASSID":"WISE-4050_4D63A1",
	"AHid":0,
	"ACnty":0,
	"ACh":6,
	"ASec":0,
	"AKey":"",
	"MAC":"00-D0-C9-F0-63-F7",
	"AIP":"192.168.1.1",
	"AMsk":"255.255.255.0",
	"AGW":"192.168.1.1",
	"DHCP":0,
	"IP":"10.0.0.100",
	"Msk":"255.255.0.0",
	"GW":"10.0.0.1",
	"IWId": "E6-F4-C6-12-C5-DE",
	"En": 0
	}

PUT

Request:
PUT /wlan_config

[Example]:
■ Request: **PUT /wlan_config**

```
Content-type: application/json
{
    "Md":0,
    "ISSID":"corega",
    "ISec":2,
    "IKey":"12345678",
    "ISSID2":"Giant",
    "ISec2":0,
    "IKey2":"",
    "ASSID":"WISE-4050_4D63A1",
    "AHid":0,
    "ACnty":0,
    "ACh":6,
    "ASec":0,
    "AKey":"",
    "DHCP":0,
    "IP":"10.0.0.100",
    "Msk":"255.255.0.0",
    "GW":"10.0.0.1",
    "IWId": "E6-F4-C6-12-C5-DE",
    "En": 0
}
```

Response: 200 OK

PATCH

Request: **PATCH /wlan_config**

[Example]:
■ Request: **PATCH /wlan_config**

```
Content-type: application/json
{
    "IKey":"advantech"
}
```

Response: 200 OK

■

Field	Abbreviation	Data Type	Property	Description
WLAN Operation Mode	Md	Number	RW	The WLAN mode of operation

0	Infrastructure Mode
1	Reserved
2	AP Mode

Field	Abbreviation	Data Type	Property	Description
Country Code	ACnty	Number	RW	The country code (channel value range depends on the country).

0	US (US FCC)
1	CA (IC Canada)
2	SG (Singapore)
3	EU (ETSI)
4	AU (Australia)
5	KR (Republic Of Korea)
6	FR (France)

				7 JP (Japan) 8 CN (China)
MAC ID	MAC	String	R	MAC address. (12+5) characters, ex, "00-D0-C9-F0-63-F7"
AP Mode				
AP – The SSID of the AP mode	ASSID	String	RW	As an access point role, the Service Set Identifier (SSID) name should be configured. Max. characters: 32
AP – Enable SSID Hidden	AHid	Number	RW	As an access point role, it can choose to hide the Service Set Identifier (SSID). 1 / 0: Enable/Disable SSID hidden.
AP – Operational Channel for the AP	ACh	Number	RW	The channel value range depends on the country. 0~165 0: Auto (2.4G only) *Reference frequency table
AP – Security Type	ASec	Number	RW	The security type of the AP. 0 Security Open 5 Security WPA/WPA2 7 Security WPA3 (MPF must enabled)
AP – Security Key	AKey	String	RW	WPA/WPA2 pass phrase of the AP Max. characters: For WPA/WPA2: 8 to 63 characters For WPA3: 8 to 255 characters
AP – IP Address	AIP	String	R	IP v4 IP address applied to AP mode
AP – Subnet Mask	AMsk	String	R	IP v4 Subnet Mask applied to AP mode
AP – Gateway Address	AGW	String	R	IP v4 Gateway address applied to AP mode
AP – Management Frame Protection	MPF	Number	RW	0 Disable 1 Enable 2 Option (Capable)
Infrastructure Mode				
Infrastructure – Enable lock AP of particular BSSID	En	Number	RW	In Infrastructure mode, enable that WISE only connect to the AP with BSSID setting. 0 Disable 1 Enable
Infrastructure – The BSSID of the Access Point	IWId	String	RW	In Infrastructure mode, the BSSID of the remote access point. (12+5) characters, ex, "B8-55-10-86-56-F7"
Infrastructure – The SSID of the Access Point	ISSID	String	RW	The SSID of the remote access point. Max. characters: 32

Infrastructure – Security Type	ISec	Number	RW	The security type of the remote access point. 0: Security Open 5: Security WPA/WPA2 7: Security WPA3 (MPF must enabled) 8: EAP-PEAP 9: EAP_TLS
Infrastructure – 802.1x EAP identity	IId	String	RW	WPA2 enterprise identity of the AP, cannot have any colon in EAP identity Max. characters:255
Infrastructure – Security Key	IKey	String	RW	WPA/WPA2/WPA3 pass phrase of the AP Max. characters: For WPA/WPA2: 8 to 63 characters For WPA2 Enterprise: 1 to 255 characters For WPA3: 8 to 255 characters
Infrastructure – Management Frame Protection	IMPF	Number	RW	0: Disable 1: Enable 2: Option (Capable)
Infrastructure – The SSID of the Second Access Point	ISSID2	String	RW	The SSID of the remote access point. Max. characters: 32
Infrastructure – Security Type of the Second AP	ISec2	Number	RW	The security type of the remote access point. 0: Security Open 5: Security WPA/WPA2 7: Security WPA3 (MPF must enabled) 8: EAP-PEAP 9: EAP_TLS
Infrastructure – 802.1x EAP identity of the Second AP	IId2	String	RW	WPA2 enterprise identity of the AP, cannot have any colon in EAP identity Max. characters: 255
Infrastructure – Security Key of the Second AP	IKey2	String	RW	WPA/WPA2/WPA3 pass phrase of the AP Max. characters: For WPA/WPA2: 8 to 63 characters For WPA2 Enterprise: 1 to 255 characters For WPA3: 8 to 255 characters
Infrastructure – Management Frame Protection	IMPF2	Number	RW	0: Disable 1: Enable 2: Option (Capable)
Infrastructure – Get IP Method	DHCP	Number	RW	Get IP Method 0: static IP 1: DHCP
Infrastructure – IP Address	IP	String	RW	IP v4 IP address

Infrastructure – Subnet Mask	Msk	String	RW	IP v4 Subnet Mask
Infrastructure – Gateway Address	GW	String	RW	IP v4 Gateway address
Operational Channel for Station mode	ICh	Number	RW	The operation channel mask for infrastructure mod. 0: Auto Bit / Channel 0~13 / 1 – 14 14~21 / 36 – 64 22~32 / 100 – 140 33~37 / 149 – 165 *Reference frequency table
Roaming	Rng	Number	RW	0/1: Disable/Enable roaming function *Not supported if lock BSSID.
RSSI Threshold	Tm	Number	RW	The RSSI threshold to trigger roaming. -1 ~ -127 unit: dBm
RSSI Hysterisis	HiS	Number	RW	The hysterisis level to switch AP. 1 ~ 127 unit: dBm
Scan Interval of Roaming	Inv	Number	RW	The scan interval for roaming. 10 ~ 65535 unit: sec
WiFi WDT Mode	CWDT	Number	RW	The WDT mode of WiFi connection. 0: Disable 1: Disassociate 2: Ping
WiFi WDT Action	AMd	Number	RW	The action after WDT occurred. 0: Reset WiFi module 1: Reboot 2: Re-associate
Disassociate time	Ds	Number	RW	The disassociate time to trigger WDT. 1~65535 unit: sec
Ping interval	Per	Number	RW	The interval to ping gateway. 1~65535 unit: sec
Loss Counts	Total	Number	RW	The loss counts of ping to trigger WDT. 1~255
Ping Target	Tag	Number	RW	IP v4 IP address of target IP for ping function. Ex: 10.1.1.1
Remarks				

Table B.23: Frequency table			
Region	**2.4G**	**5G**	**5G (AP mode)**
US	1 – 11	36 – 64 100 – 140 149-165	36-48 149 – 165
CA	1 – 11	36 – 64 100 – 116 132 – 140 149 – 165	36-48 149-165
SG	1 – 11	36 – 64 100 – 140 149 – 165	36-48 149-165
EU	1 – 13	36 – 64 100 – 140	36-48
AU	1 – 13	36 – 64 100 – 140	36-48
KR	1 – 13	36 – 64 100 – 140	36-48
FR	1 – 13	36 – 64 100 – 140 149 – 165	36-48 149-165
JP	1 – 14	36 – 64 100 – 140	36-48
CN	1 – 13	149 – 165	149 – 165

B.2.22 Data Logger

/log_dataoption/slot_0

Table B.24: Data Logger	
Description	Retrieves the data options for data logger.
URL Structure	**http://10.0.0.1/log_dataoption**
HTTP Method	GET: Returns the representation of all of logged data type resource. PUT: Replace all of logged data type resource. PATCH: Apply partial modifications to logged data type resource.

GET	**Request:**
	GET /log_dataoption
	
	[Example]:
	■ Request: **GET /log_dataoption for ADAM-T160 module**
	
	Content-type: application/json
	Response: 200 OK
	{
	" TIM ":1,
	"DIChM":15,
	"DOChM":1
	}
	
	■ Request: **GET /log_dataoption for ADAM-3600-A1F module**
	
	Content-type: application/json
	Response: 200 OK
	{
	" TIM ":1,
	"DIChM":[65535,0,15,0,0],
	"DOChM":[255,0,0,3,0],
	"AIChM":[0,7,0,0,0],
	"AOChM":[0,0,0,0,0],
	"AILgD":193,
	"AOLgD":0,
	"DILgD": 0,
	"SysLgD":1
	}

PUT	Request:
	PUT / log_dataoption
	
	[Example]:
	■ Request: **PUT / log_dataoption for ADAM-T160 module**
	Content-type: application/json
	{
	" TIM ":1,
	"DIChM":3,
	"DOChM":7
	}
	Response: 200 OK

PATCH	Request: **PATCH / log_dataoption**
	
	[Example]:
	■ Request: **PATCH / log_dataoption**
	Content-type: application/json
	{
	"TIM":0
	}
	
	Response: 200 OK

■ Resource value definitions:

Field	Abbreviation	Data Type	Property	Description
Timestamp	TIM	Number	R	1 : the storage of timestamp is always enabled.
DI Enabled Channel Mask	DIChM	Number	RW	Mask for DI enabled channels of all slots.
DO Enabled Channel Mask	DOChM	Number	RW	Mask for DO enabled channels of all slots.
AI Enabled Channel Mask	AIChM	Number	RW	Mask for AI enabled channels of all slots. Total channels = AI channel number + 1 average channel
AO Enabled Channel Mask	AOChM	Number	RW	Mask for AO enabled channels of all slots.
AI log data options	AILgD	Number	RW	Data types enable mask for AI measurement. 1 or 0 of each bit: Enable or Disable the type Notice: AI status always be logged when AI data in logging.
AO log data options	AOLgD	Number	RW	Data types enable mask for AO measurement. 1 or 0 of each bit: Enable or Disable the type

Bit	Description
0	AI raw value
1	AI engineering value
2	AI Max raw value
3	AI Max. engineering value
4	AI Min. raw value
5	AI Min. engineering value
6	AI value after scaling
7	AI status (always enabled)
8	AI physical value
9~	(Reserved)

Bit	Description
0	AO raw value
1	AO engineering value
2	AO value after scaling
3	AOstatus
4~	(Reserved)

<table>
<tr><td>Sensor Enabled Channel Mask</td><td>SChM</td><td>Number</td><td>RW</td><td>Mask for sensor enabled channels of all slots.
Total channels = sensor channel number</td></tr>
<tr><td>Sensor log data options</td><td>SLgD</td><td>Number</td><td>RW</td><td>Data types enable mask for sensor measurement.
1 or 0 of each bit: Enable or Disable the type
Notice: Sensor status always be logged when sensor data in logging.
<table>
<tr><td>Bit</td><td>Description</td></tr>
<tr><td>0</td><td>Sensor engineering value</td></tr>
<tr><td>1</td><td>Sensor Max. engineering value</td></tr>
<tr><td>2</td><td>Sensor Min. engineering value</td></tr>
<tr><td>3</td><td>Sensor status (always enabled)</td></tr>
<tr><td>4</td><td>Sensor high alarm</td></tr>
<tr><td>5</td><td>Sensor low alarm</td></tr>
<tr><td>6~</td><td>(Reserved)</td></tr>
</table></td></tr>
<tr><td>Stacklight Sensor Enabled Channel Mask</td><td>LSChM</td><td>Number</td><td>RW</td><td>Mask for stacklight sensor enabled channels of all slots.
Total channels = stacklight channel number</td></tr>
<tr><td>Stacklight log data options</td><td>LSLgD</td><td>Number</td><td>RW</td><td>Data types enable mask for light sensor measurement.
1 or 0 of each bit: Enable or Disable the type
Notice: Light sensor status always be logged when sensor data in logging.
<table>
<tr><td>Bit</td><td>Description</td></tr>
<tr><td>0</td><td>State</td></tr>
<tr><td>1</td><td>Illumination</td></tr>
<tr><td>2</td><td>Frequency</td></tr>
<tr><td>3</td><td>State count</td></tr>
<tr><td>4</td><td>State total time</td></tr>
<tr><td>5~</td><td>(Reserved)</td></tr>
</table></td></tr>
<tr><td colspan="5">Remarks</td></tr>
</table>

B.2.23 Logging Conditions

Table B.25: Logging conditions	
Description	Retrieves the configuration for data logger.
URL Structure	**http://10.0.0.1/log_control**
HTTP Method	GET: Returns the representation of all of logged data configuration resource. PUT: Replace all of logged data configuration resource. PATCH: Apply partial modifications to logged data configuration resource.

GET	Request:
	GET /log_control
	
	[Example]:
	■ Request: **GET /log_control for ADAM-T160 module**
	
	Content-type: application/json
	Response: 200 OK
	{
	“Per”:0,
	“CWDT”:0,
	“Pitv”:600,
	“DICOS”:15,
	“DOCOS”:0,
	“Cir”:1,
	“Rst”:0
	}
	
	■ Request: **GET /log_control for ADAM-3600-A1F module**
	
	Content-type: application/json
	Response: 200 OK
	{
	“Per”:0,
	“CWDT”:0,
	“Pitv”:600,
	“DICOS”:[65535,0,0,0,7],
	“DOCOS”:[15,0,0,8,0],
	“AICOS”:[0,3,0,0,0],
	“AOCOS”:[0,0,0,0,0],
	“AIDR”:10,
	“AODR”:5,
	“Cir”:1,
	“Rst”:0,
	“IntLg”:1,
	“SDLg”:1,
	“USBLg”:0,
	“SDF”:0,
	“USBF”:0
	}

PUT	Request: **PUT / log_control**

	[Example]:
	■ Request: **PUT / log_control for ADAM-T160 module**
	Content-type: application/json
	{
	“Per”:0,
	“CWDT”:1,
	“Pitv”:600,
	“DICOS”:15,
	“DOCOS”:0,
	“Cir”:1,
	“Rst”:0
	}
	Response: 200 OK

PATCH	Request: **PATCH / log_control**
	
	[Example]:
	■ Request: **PATCH / log_control**
	Content-type: application/json
	{
	“Per”:0
	}
	
	Response: 200 OK

■ Resource value definitions:

Field	Abbreviation	Data Type	Property	Description
Enable Periodical Log	Per	Number	RW	1 or 0: Enable or Disable the periodical log
Enable Communi-cation WDT Trigger	CWDT	Number	RW	1 or 0: Enable or Disable the logging triggered by communication WDT. **Not use for private server
Period Interval	Pitv	Number	RW	Periodical log Time Interval (unit: 0.1s) Default: 600 (1 minute) * Normal Periodical logging * Logging after communication fail occurred.
DI Change of state Enabled Channel Mask	DICOS	Number	RW	Mask for DI enabled channels of all slots.
DO Change of state Enabled Channel Mask	DOCOS	Number	RW	Mask for DO enabled channels of all slots.
AI Deviation Enabled Channel Mask	AICOS	Number	RW	Mask for AI enabled channels of all slots. Total channels = AI channel number + 1 average channel
AO Deviation Enabled Channel Mask	AOCOS	Number	RW	Mask for AO enabled channels of all slots.
AI Deviation Rate	AIDR	Number	RW	AI deviation value. 1 ~ 99 % FSR (Full Scale Range) (Unit: %) Default: 5 (%)
AO Deviation Rate	AODR	Number	RW	AI deviation value. 1 ~ 99 % FSR (Full Scale Range) (Unit: %) Default: 5 (%)

Sensor Deviation Enabled Channel Mask	SCOS	Number	RW	Mask for sensor enabled channels of all slots.
Sensor high alarm	SHiA	Number	RW	Mask for sensor high alarm enabled channels of all slots.
Sensor low alarm	SLoA	Number	RW	Mask for sensor low alarm enabled channels of all slots.
Stack light sensor change of state channel mask	LSCOS	Number	RW	Mask for stack light sensor enabled channels of all slots.
Enable Circular Operation for Built-in Memory	Cir	Number	RW	1 or 0: Enable or Disable the Circular Operation for Built-in Memory Default: 6 **Not use for private server
Reset Built-in Log once System Restart	Rst	Number	RW	1 or 0: Enable or Disable to reset the Built-in log when the system powers on. **Not use for private server
Remarks				

B.2.24 Cloud Connections

Table B.26: Cloud Connections	
Description	Retrieves the Cloud configuration
URL Structure	**http://10.0.0.1/cloud_config**
HTTP Method	GET: Returns the representation of all of Cloud configuration resource. PUT: Replace all of Cloud configuration resource. PATCH: Apply partial modifications to WISE Cloud configuration resource.
GET	Request: GET /cloud_config [Example]: ?Request : GET /cloud_config Content-type: application/json Response: 200 OK { "Sel":0, "Code":"", "PWeb":8080, "SSLEn":0, "IP":"10.0.0.1", "Pauth":1, "Pu":"admin", "Pw":"password", "Uurl":"/private_server_upload/", "Durl":"/private_server_push/io_data/", "Surl":"/private_server_push/sys_event/" "Nm":"test", "En":"oss-cn-shanghai.aliyuncs.com", "Id":"j6Ob8RXgaETRk66d", "AKey":"fR6fY2Z4hGBFav6GhVGm9rjPRxwdc", "FCS": 119 } ChungHua Telelcom IoT: Response: 200 OK

	{"Sel":134,"FCS":7952,"Id":" 11358917705","AKey":" DKGXXEK-FA7XM14RF14","HbF":0,"PubQ":0,"SubQ":0}

PUT	Request:
	PUT /cloud_config
	
	[Example]:
	■ Request: **PUT /cloud_config**
	Content-type: application/json
	{
	"Sel":0,
	"Code":"GMers8WXZ4wAAAAAAAAUnlukVo8fJMZ_upibsczxs9A",
	"PWeb":8080,
	"SSLEn":0,
	"IP":"10.0.0.1",
	"Pauth":1,
	"Pu":"admin",
	"Pw":"password",
	"Uurl":"/private_server_upload/",
	"Durl":"/private_server_push/io_data/",
	"Surl":"/private_server_push/sys_event/"
	"Nm":"test",
	"En":"oss-cn-shanghai.aliyuncs.com",
	"Id":"j6Ob8RXgaETRk66d",
	"AKey":"fR6fY2Z4hGBFav6GhVGm9rjPRxwdc"
	}
	
	Response: 200 OK
	
	ChungHua Telelcom IoT:
	{"Sel":134,"FCS":7952,"Id":" 11358917705","AKey":" DKGXXEK-FA7XM14RF14","HbF":0,"PubQ":0,"SubQ":0}
	
	Response: 200 OK

PATCH	Request: **PATCH /cloud_config**
	
	[Example]:
	■ Request: **PATCH /cloud_config**
	Content-type: application/json
	{
	" Sel ":1
	}
	
	Response: 200 OK

■ Resource value definitions:

Field	Abbreviation	Data Type	Property	Description
Cloud Server Select	Sel	Number	RW	The cloud server select

Index	Description
0	Disabled
1	Dropbox
2	Aliyun OSS
4	Private Server
132	LwM2M
134	ChungHwa Telecom

<table>
<tr><td></td><td></td><td></td><td></td><td>Others is read-only and can refer to Cloud Service Select Table (WISE-4000): all supported cloud in Cloud Service Support Table can RW
Note: This Field should be the first key/ value pair in JSON object</td></tr>
<tr><td>Cloud Server support</td><td>FCS</td><td>Number</td><td>R</td><td>Supported cloud functions, refer to Cloud Service Support Table</td></tr>
<tr><td colspan="5">Private server</td></tr>
<tr><td>Private server port</td><td>PWeb</td><td>Number</td><td>RW</td><td>Private server port
Value: 1~65535</td></tr>
<tr><td>Enable SSL secure</td><td>SSLEn</td><td>Number</td><td>RW</td><td>Enable SSL secure of private server<table><tr><td>0</td><td>Disable</td></tr><tr><td>1</td><td>Enable</td></tr><tr><td>2</td><td>Reserved</td></tr></table></td></tr>
<tr><td>Private server ip</td><td>IP</td><td>String</td><td>RW</td><td>Set private/Azure server ip address or Host Name.
Max. characters: 128</td></tr>
<tr><td>Enable private server authorization</td><td>Pauth</td><td>Number</td><td>RW</td><td>Enable authorization<table><tr><td>0</td><td>Disable</td></tr><tr><td>1</td><td>Basic Authorization</td></tr><tr><td>2</td><td>Reserved</td></tr></table></td></tr>
<tr><td>Authorization user name</td><td>Pu</td><td>String</td><td>RW</td><td>Set authorization user name
Max. characters: 64</td></tr>
<tr><td>Authorization password</td><td>Pw</td><td>String</td><td>RW</td><td>Set authorization password
Max. characters: 16</td></tr>
<tr><td>Private Server Upload web url</td><td>Uurl</td><td>String</td><td>RW</td><td>Set Private Server Upload web url
Max. characters: 128</td></tr>
<tr><td>Push I/O data web url</td><td>Durl</td><td>String</td><td>RW</td><td>Set push I/O data web url
Max. characters: 128</td></tr>
<tr><td>Push System event url</td><td>Surl</td><td>String</td><td>RW</td><td>Set push System event web url
Max. characters: 128</td></tr>
<tr><td colspan="5">iSensingMQTT</td></tr>
<tr><td>iSensingMQTT/ Cloud Host Name</td><td>Nm</td><td>String</td><td>RW</td><td>Set the host name of iSensingMQTT/ Cloud
Max. characters: 64
Ex, "iot.eclipse.org"</td></tr>
<tr><td>iSensingMQTT/ Cloud Host port</td><td>PWeb</td><td>Number</td><td>RW</td><td>iSensingMQTT/Cloud port
Value: 1~65535</td></tr>
<tr><td>Enable iSensing-MQTT/Cloud SSL secure</td><td>SSLEn</td><td>Number</td><td>RW</td><td>Enable SSL secure of iSensingMQTT/ Cloud<table><tr><td>0</td><td>Disable</td></tr><tr><td>1</td><td>Enable</td></tr></table></td></tr>
<tr><td>iSensingMQTT/ Cloud User Name</td><td>Pu</td><td>String</td><td>RW</td><td>Set the user name of iSensingMQTT/ Cloud<table><tr><td>Model</td><td>Max. characters</td></tr><tr><td>WISE-4220</td><td>192</td></tr></table></td></tr>
</table>

iSensingMQTT/ Cloud User Password	Pw	String	RW	Set the user password of iSensingMQTT/Cloud Model: Max. characters WISE-4220: 192
iSensingMQTT/ Cloud Heartbeat Frequency	HbF	Number	RW	The time interval of iSensingMQTT/ Cloud heartbeat packet. Value: 0~65535
iSensingMQTT/ Cloud Publish QoS	PubQ	Number	RW	Set publish QoS of iSensingMQTT/ Cloud Value: 0~2
iSensingMQTT/ Cloud Subscribe QoS	SubQ	Number	RW	Set subscribe QoS of iSensingMQTT/ Cloud Value: 0~2
Enable iSensingMQTT/Cloud websocket	SEn	Number	RW	Enable websocket of iSensingMQTT/ Cloud 0: Disable 1: Enable Default: 1
WebSocket path	Surl	String	RW	WebSocket authorization path Model: Max. characters WISE-4471, WISE-4671, WISE-4210-AP: 64 WISE-4000/Wi-Fi, WISE-4000/LAN: 19
Enable iSensingMQTT/Cloud Publish Retain	PE	Number	RW	Enable Publish Retain of iSensingMQTT/Cloud 0: Disable 1: Enable Default: 1
Enable iSensingMQTT/Cloud Will Retain	We	Number	RW	Enable Will Retain of iSensingMQTT/ Cloud 0: Disable 1: Enable Default: 1
DI Data Recovery	DIChM	Number	RW	DI Data Recovery Channel Enable Mask
DO Data Recovery	DOChM	Number	RW	DO Data Recovery Channel Enable Mask
AI Data Recovery	AIChM	Number	RW	AI Data Recovery Channel Enable Mask
Sensor Data Recovery	SChM	Number	RW	Sensor Data Recovery Channel Enable Mask
Stack Light Sensor Data Recovery	LSChM	Number	RW	Stack Light Sensor Data Recovery Channel Enable Mask
COM1 Data Recovery	Msk	Number	RW	RTU COM1 rule Data Recovery rule Enable Mask

COM2 Data Recovery	AMsk	Number	RW	RTU COM2 rule Data Recovery rule Enable Mask
Azure HTTP				
Cloud Server Select	Sel	Number	RW	The cloud server select Index: Description None: Disabled 64: Azure Others is read-only and refer to Cloud Service Select Table WISE-4000 has no this resource.
Azure Connection String	SAS	String	RW	Azure Connection String Max. characters: 192
Dropbox				
User Code	Code	String	RW	Set Oauth user code for WISE device generate access token Max. characters: 64
IFTTT				
Event Name	Evt	String	RW	Webhooks event name Max. 32 characters inclusive of a null-end character (i.e. up to 31 characters)
Webhooks Key	AKey	String	RW	Set Webhooks key Max. characters: 48
ChungHua Telelcom IoT				
CHT device id	Id	String	RW	Set the device id Max. characters: 63 + 1
CHT device key	AKey	String	RW	Set the device key Max. characters: 63 + 1
CHT heartbeat	HbF	Number	RW	Set the time interval of heartbeat Model: Value (Unit: second) WISE-4471: 0 and 10 ~ 65535 Others: 0 ~ 65535
CHT Publish QoS	PubQ	Number	RW	Set the Publish QoS Value: 0~2
CHT Subscribe QoS	SubQ	Number	RW	Set the Subscribe QoS Value: 0~2
DI Data Recovery	DIChM	Number	RW	DI Data Recovery Channel Enable Mask
DO Data Recovery	DOChM	Number	RW	DO Data Recovery Channel Enable Mask
AI Data Recovery	AIChM	Number	RW	AI Data Recovery Channel Enable Mask
Sensor Data Recovery	SChM	Number	RW	Sensor Data Recovery Channel Enable Mask
Stack Light Sensor Data Recovery	LSChM	Number	RW	Stack Light Sensor Data Recovery Channel Enable Mask
COM1 Data Recovery	Msk	Number	RW	RTU COM1 rule Data Recovery rule Enable Mask
COM2 Data Recovery	AMsk	Number	RW	RTU COM2 rule Data Recovery rule Enable Mask
Azure MQTT				

Azure Connection String	SAS	String	RW	Azure Connection String Max. characters: 192
Azure Host port	PWeb	Number	RW	Value: 1~65535
Azure Heartbeat Frequency	HbF	Number	RW	The time interval of heartbeat packet. Model: WISE-4471 — Value (Unit: second): 0 and 10 ~ 65535
Azure Publish QoS	PubQ	Number	RW	Set publish QoS Value: 0~1
Azure Subscribe QoS	SubQ	Number	RW	Set subscribe QoS Value: 0~1
Enable Azure websocket	SEn	Number	RW	Enable websocket 0 Disable 1 Enable Default: 0
DI Data Recovery	DIChM	Number	RW	DI Data Recovery Channel Enable Mask
DO Data Recovery	DOChM	Number	RW	DO Data Recovery Channel Enable Mask
AI Data Recovery	AIChM	Number	RW	AI Data Recovery Channel Enable Mask
Sensor Data Recovery	SChM	Number	RW	Sensor Data Recovery Channel Enable Mask
Stack Light Sensor Data Recovery	LSChM	Number	RW	Stack Light Sensor Data Recovery Channel Enable Mask
COM1 Data Recovery	Msk	Number	RW	RTU COM1 rule Data Recovery rule Enable Mask
COM2 Data Recovery	AMsk	Number	RW	RTU COM2 rule Data Recovery rule Enable Mask
Remarks				

B.2.25 iSensing MQTT/Cloud Configuration

Table B.27: iSensing MQTT/Cloud Configuration	
Description	Retrieves the iSensingMQTT/Cloud configuration
URL Structure	**http://10.0.0.1/iSensingMQTT_config**
HTTP Method	GET: Returns the representation of all of iSensingMQTT/Cloud configuration resource. PUT: Replace all of iSensingMQTT/Cloud configuration resource. PATCH: Apply partial modifications to WISE iSensingMQTT/Cloud configuration resource.
GET	Request: **GET /iSensingMQTT_config** [Example]: ■ Request: **GET /iSensingMQTT_config** Content-type: application/json Response: 200 OK { "Sel": 16, "Nm": "172.16.12.252", "PWeb": 8080,
	"SSLEn": 0, "Pu": "admin", "Pw": "admin", "HbF": 60, "PubQ": 2, "SubQ": 1, "SEn": 0, "Surl":"mqtt", " PE ":"1", " We ":"1" }

PUT	Request:
	PUT /iSensingMQTT_config
	
	[Example]:
	■ Request: **PUT /iSensingMQTT_config**
	Content-type: application/json
	{
	"Sel": 16,
	"Nm": "172.16.12.252",
	"PWeb": 8080,
	"SSLEn": 0,
	"Pu": "admin",
	"Pw": "admin",
	"HbF": 60,
	"PubQ": 2,
	"SubQ": 1,
	"SEn": 0,
	"Surl":"mqtt"
	" PE ":"1",
	" We ":"1"
	}
	Response: 200 OK

PATCH	Request: **PATCH /iSensingMQTT_config**
	
	[Example]:
	■ Request: **PATCH /iSensingMQTT_config**
	Content-type: application/json
	{
	" HbF ":10
	}
	
	Response: 200 OK

■ Resource value definitions:

Field	Abbreviation	Data Type	Property	Description
Cloud Server Select	Sel	Number	RW	The cloud server select Index: Description None: Disabled 16: iSensing MQTT Others is read-only and refer to Cloud Service Select Table WISE-4000 has no this resource.
iSensingMQTT/ Cloud Host Name	Nm	String	RW	Set the host name of iSensingMQTT/ Cloud Max. characters: 64 Ex, "iot.eclipse.org"
iSensingMQTT/ Cloud Host port	PWeb	Number	RW	iSensingMQTT/Cloud port Value: 1~65535
Enable iSensing-MQTT/Cloud SSL secure	SSLEn	Number	RW	Enable SSL secure of iSensingMQTT/ Cloud 0: Disable 1: Enable
iSensingMQTT/ Cloud User Name	Pu	String	RW	Set the user name of iSensingMQTT/ Cloud WISE-4471,WISE-4671, WISE-4210-AP, WISE-4000/Wi-Fi, WISE-4470,WISE-4670: 128 Others: 64
iSensingMQTT/ Cloud User Pass-word	Pw	String	RW	Set the user password of iSensing-MQTT/Cloud Model: Max. characters

				WISE-4471, WISE-4671, WISE-4210-AP, WISE-4000/Wi-Fi, WISE-4470,WISE-4670: 128 Others: 64
iSensingMQTT/ Cloud Heartbeat Frequency	HbF	Number	RW	The time interval of iSensingMQTT/ Cloud heartbeat packet. Model: Value (Unit: second) WISE-4471, WISE-4671: 0 and 10 ~ 65535 Others: 0 ~ 65535
iSensingMQTT/ Cloud Publish QoS	PubQ	Number	RW	Set publish QoS of iSensingMQTT/ Cloud Value: 0~2
iSensingMQTT/ Cloud Subscribe QoS	SubQ	Number	RW	Set subscribe QoS of iSensingMQTT/ Cloud Value: 0~2
Enable iSensing-MQTT/Cloud websocket	SEn	Number	RW	Enable websocket of iSensingMQTT/ Cloud 0: Disable 1: Enable Default: 1
WebSocket path	Surl	String	RW	WebSocket authorization path Model: Max. characters WISE-4471, WISE-4671, WISE-4210-AP, WISE-4470,WISE-4670: 64 WISE-4000/Wi-Fi, WISE-4000/ LAN: 19
Enable iSensing-MQTT/Cloud Publish Retain	PE	Number	RW	Enable Publish Retain of iSensing-MQTT/Cloud 0: Disable 1: Enable Default: 1
Enable iSensing-MQTT/Cloud Will Retain	We	Number	RW	Enable Will Retain of iSensingMQTT/ Cloud 0: Disable 1: Enable Default: 1

SSL Certificate Type	CtS	Number	RW	SSL/TLS Certificate Type 0: CA signed server certificate 1: Selfed signed certificate (CA file, cert) 2: Reserved for CA certificate only(only cert file) Default: 0 Note: WISE-4000 only
Enable Customized Topics and JSON Keys	CtOv	Number	RW	Enable Customized Topics and JSON Keys of iSensingMQTT/Cloud 0: Disable 1: Enable Default: 0 Note: WISE-4000 only
DI Data Recovery	DIChM	Number	RW	DI Data Recovery Channel Enable Mask
DO Data Recovery	DOChM	Number	RW	DO Data Recovery Channel Enable Mask
AI Data Recovery	AIChM	Number	RW	AI Data Recovery Channel Enable Mask
COM1 Data Recovery	Msk	Number	RW	RTU COM1 rule Data Recovery rule Enable Ma
COM2 Data Recovery	MCh	Number	RW	RTU COM2 rule Data Recovery rule Enable Mask
Position Data Recovery	Per	Number	RW	Position Data Recovery rule Enable
Enable Restful over MQTT	Rst	Number	RW	Enable Restful over MQTT of iSensingMQTT/Cloud 0: Disable 1: Enable Default: 0
Remarks				

B.2.26 P2P Mode & Common Configurations

Table B.28: P2P Mode & Common Configurations

Description	Retrieves information about the common Peer-to-Peer configuration.
URL Structure	**http://10.0.0.1/p2p_mode**
HTTP Method	GET: Returns the representation of all of common Peer-to-Peer configuration resource. PUT: Replace all of common Peer-to-Peer configuration resource. PATCH: Apply partial modifications to common Peer-to-Peer configurations resource.

GET	Request:
	GET /p2p_mode
	
	[Example]:
	■ Request: **GET /p2p_mode**
	
	Content-type: application/json
	Response: 200 OK
	{
	"Md": 0,
	"En": 1,
	"QL": 0,
	"ECS": 1,
	"PpG": 5048
	}

PUT	Request:
	PUT /p2p_mode
	
	[Example]:
	■ Request: **PUT /p2p_mode**
	Content-type: application/json
	{
	"Md": 1,
	"En": 1,
	"QL": 0,
	"ECS": 1,
	"PpG": 5048
	}
	Response: 200 OK

PATCH	Request:
	PATCH /p2p_mode
	
	[Example]:
	■ Request: **PATCH /p2p_mode**
	Content-type: application/json
	{
	"En": 0
	}
	
	Response: 200 OK

■ Resource value definitions:

Field	Abbreviation	Data Type	Property	Description
P2P Mode	Md	Number	RW	Peer-to-Peer mode. 0 P2P Disabled

				1 Basic Mode 2 Advanced Mode
Enable P2P Periodically Transmission	En	Number	RW	1 or 0: Enable or Disable P2P periodically transmission Default: 1
QoS level	QL	Number	RW	QoS level for response 0: no response 1: retry for response Default: 0
Encryption type	ECS	Number	RW	Encryption type for transmission package 0: no encryption 1: AES-128 Default: 0
P2P Destination Port	PpG	Number	RW	1~65534: Destination port number for P2P remote message packets Default: 5048
Remarks				

B.2.27 P2P Destination IP Table

Table B.29: P2P Destination IP Table

Description	Retrieves information about the destination table.
URL Structure	**http://10.0.0.1/p2p_iptable** **http://10.0.0.1/p2p_iptable/idx_index**
HTTP Method	GET: Returns the representation of all the destination table resource. PUT: Replace all the destination table resource. PATCH: Apply partial modifications to the destination table resource.
GET	Multi Item Request: **GET /p2p_iptable** Single Item Request: **GET /p2p_iptable/idx_index** [Example]: ■ Request: **GET /p2p_iptable** Content-type: application/json Response: 200 OK { "IpTbl": [{ "Idx": 0, "IP": "172.18.3.5", "Id": 27488, "Pw":"0F0F0F0F0F0F0F0F" }, { "Idx": 1, "IP": "172.18.3.5", "Id": 27472,

GET (Cont.)	"Pw":"0F0F0F0F0F0F0F0F"
	},
	{
	"Idx": 2,
	"IP": "255.255.255.255",
	"Id": 27488,
	"Pw":"0F0F0F0F0F0F0F0F"
	},
	{
	"Idx": 3,
	"IP": "255.255.255.255",
	"Id": 27488,
	"Pw":"0F0F0F0F0F0F0F0F"
	},
	{
	"Idx": 4,
	"IP": "255.255.255.255",
	"Id": 27488,
	"Pw":"0F0F0F0F0F0F0F0F"
	},
	{
	"Idx": 5,
	"IP": "255.255.255.255",
	"Id": 27488,
	"Pw":"0F0F0F0F0F0F0F0F"
	},
	{
	"Idx": 6,
	"IP": "255.255.255.255",
	"Id": 27488,
	"Pw":"0F0F0F0F0F0F0F0F"
	},
	……
	……
	{
	"Idx": 15,
	"IP": "255.255.255.255",
	"Id": 27488,
	"Pw ":"0F0F0F0F0F0F0F0F"
	}
	]
	}
	
	■ Request: **GET /p2p_iptable/idx_5**
	
	Content-type: application/json
	Response: 200 OK
	{
	"Idx": 5,
	"IP": "192.168.1.2",
	"Id": 27488,
	"Pw ":"0F0F0F0F0F0F0F0F"
	}

PUT	Single/Multi Item Request: **PUT /p2p_iptable** Single Item Request: **PUT /p2p_iptable/idx_index** [Example]:

PUT (Cont.)	■ Request: **PUT /p2p_iptable**
	Content-type: application/json
	{
	"IpTbl": [
	{
	"Idx": 0,
	"IP": "172.18.3.5",
	"Id": 27488,
	"Pw ":"0F0F0F0F0F0F0F0F"
	},
	{
	"Idx": 1,
	"IP": "172.18.3.5",
	"Id": 27472,
	"Pw ":"0F0F0F0F0F0F0F0F"
	},
	{
	"Idx": 2,
	"IP": "255.255.255.255",
	"Id": 27488,
	"Pw ":"0F0F0F0F0F0F0F0F"
	},
	{
	"Idx": 3,
	"IP": "255.255.255.255",
	"Id": 27488,
	"Pw ":"0F0F0F0F0F0F0F0F"
	},
	{
	"Idx": 4,
	"IP": "255.255.255.255",
	"Id": 27488,
	"Pw ":"0F0F0F0F0F0F0F0F"
	},
	{
	"Idx": 5,
	"IP": "255.255.255.255",
	"Id": 27488,
	"Pw ":"0F0F0F0F0F0F0F0F"
	},
	{
	"Idx": 6,
	"IP": "255.255.255.255",
	"Id": 27488,
	"Pw ":"0F0F0F0F0F0F0F0F"
	},
	……
	……
	{
	"Idx": 15,
	"IP": "255.255.255.255",
	"Id": 27488,
	"Pw ":"0F0F0F0F0F0F0F0F"
	}
	]
	}
	
	Response: 200 OK

<table>
<tr><td>PUT (Cont.)</td><td>■ Request: PUT /p2p_iptable/idx_3
Content-type: application/json
{
"Idx": 3,
"IP": "172.18.3.66",
"Id": 27488
}
Response: 200 OK</td></tr>
<tr><td>PATCH</td><td>Single/Multi Item Request:
PATCH /p2p_iptable
Single Item Request:
PATCH /p2p_iptable/idx_index

[Example]:
■ Request: PATCH /p2p_iptable
Content-type: application/json
{
" IpTbl ": [
{
"Idx": 3,
"IP": "172.18.3.66"
},
{
"Idx": 5,
"IP": "172.18.3.6",
"Id": 27666
},
{
"Idx": 6,
"IP": "172.18.3.45"
}
]
}

Response: 200 OK

■ Request: PATCH /p2p_iptable/idx_3
Content-type: application/json
{
"IP": "172.18.3.45"
}

Response: 200 OK</td></tr>
</table>

■ JSON array name definition:

Field	Abbreviation	Data Type
Array of IP table configurations	IpTbl	Array

■ Resource value definitions:

Field	Abbreviation	Data Type	Property	Description
Item Index	Idx	Number	R	0, 1, …15: Index number of IP set
Destination IP Address	IP	String	RW	The array of IP v4 Destination IP address to receive the P2P command packet Max array number: 16
Destination model ID	Id	Number	RW	The array of module ID. Max array number: 16

Module Name	Module ID	
	Hex	decimal
WISE-4012	0x6B 0x12	27410
WISE-4050	0x6B 0x50	27472
WISE-4051	0x6B 0x51	27473
WISE-4060	0x6B 0x60	27488
WISE-4012E	0x6C 0x12	27666
WISE-4010/ LAN	0x6A 0x10	27152
WISE-4050/ LAN	0x6A 0x50	27216
WISE-4060/ LAN	0x6A 0x60	27232
WISE-4250-S250	0x42 0x50 0x02 0x50	1,112,539,728

Destination PW	Pw	String	RW	Root PW *Perform a XOR(^) operation on password and the constant value, 0x3F. Example, password '00000000'->"0F0F0F0F0F0F0F0F"

Remarks

B.2.28 P2P Basic Mode Configurations

Table B.30: P2P Basic Mode Configurations

Description	Retrieves information about the Peer-to-Peer Basic mode configuration.
URL Structure	**http://10.0.0.1/p2p_basic**
HTTP Method	GET: Returns the representation of all of Peer-to-Peer Basic mode configuration resource. PUT: Replace all of Peer-to-Peer Basic mode configuration resource. PATCH: Apply partial modifications to Peer-to-Peer Basic mode configurations resource.

GET	Request:
	GET /p2p_basic
	
	[Example]:
	■ Request: **GET /p2p_basic, ADAM-3600-A1F for example**
	
	Content-type: application/json
	Response: 200 OK
	{
	"DIChM ": [15,0,0,0,0],
	"AIChM": [0,0,15,0,0],
	"Inv": [15,0,0,0,0],
	"COS": 1,
	"Dev": 5,
	"PItv": 3,
	"IP": 2,
	"Id": 27488,
	"BiUn": [0, 0, 15, 0, 0]
	}
	
	[Example]:
	■ Request: **GET /p2p_basic, WISE-4012E for example**
	
	Content-type: application/json
	Response: 200 OK
	{
	"DIChM": [3],
	"AIChM": [3],
	"Inv": [1],
	"COS": 1,
	"Dev": 5,
	"PItv": 3,
	"IP": 3,
	"BiUn": [2]
	}

PUT	Request:
	PUT /p2p_basic
	
	[Example]:
	■ Request: **PUT /p2p_basic, ADAM-3600-A1F for example**
	Content-type: application/json
	{
	"DIChM": [3,0,0,0,0],
	"AIChM": [0,0,0,3,0],
	"Inv": [1,0,0,0,0],
	"COS": 1,
	"Dev": 5,
	"PItv": 3,
	"IP":3,
	"BiUn": [0, 0, 0, 2, 0]
	}
	Response: 200 OK
	
	[Example]:
	■ Request: **PUT /p2p_basic, WISE-4012E for example**
	Content-type: application/json
	{
	"DIChM": [3],
	"AIChM": [1],
	"Inv": [3],
	"COS": 1,
	"Dev": 5,
	"PItv": 3,
	"IP": 3,
	"BiUn": [1]
	}
	Response: 200 OK

PATCH	Request:
	PATCH /p2p_basic
	
	[Example]:
	■ Request: **PATCH /p2p_basic**
	Content-type: application/json
	{
	"COS": 0
	}
	
	Response: 200 OK

■ Resource value definitions:

Field	Abbreviation	Data Type	Property	Description
Enabled DI Channel Mask	DIChM	Array	RW	Mask for DI Basic Peer-to-Peer enabled channels of all slots. The mask is including of DI and DI mode in UI. *The index is sort by the I/O type. DI is first. For example: If the module is 2DI and 2UI, the DI mask is

				Mask Bit / Channel 0 / DI_0 1 / DI_1 2 / UI_0 3 / UI_1
Enabled AI Channel Mask	AIChM	Array	RW	Mask for AI Basic Peer-to-Peer enabled channels of all slots. The mask is including of AI and AI mode in UI. *The index is sort by the I/O type. AI is first. For example: If the module is 2AI and 2UI, the AI mask is Mask Bit / Channel 0 / AI_0 1 / AI_1 2 / UI_0 3 / UI_1
Invert Signal	Inv	Array	RW	Mask for DI invert channels of all slots (Only for DI)
Enable DI change of state or AI deviation	COS	Number	RW	1 or 0: Enable or Disable DI change of state or AI deviation.
AI Deviation value	Dev	Number	RW	1 ~ 100: AI deviation value. (Unit: %)
Period Time	PItv	Number	RW	Period Time (unit: sec): 0-86400 Default: 5 sec 0: disable
Destination IP Table Mask	IP	Number	RW	The mask of destination IP table Mask Bit / IP table index 0 / IP 0 1 / IP 1 2 / IP 2 3 / IP 3 4 / IP 4 5 / IP 5 6 / IP 6 7 / IP 7 8 / IP 8 9 / IP 9 10 / IP 10 11 / IP 11 12 / IP 12 13 / IP 13 14 / IP 14 15 / IP 15

AI Channel Mask of Bi-polar to Uni-polar conversion	BiUn	Array	RW	AI Channel Mask of Bi-polar to Uni-polar conversion for all slots.(Only for AI)
Remarks				

B.2.29 P2P Advanced Mode Configurations – AI mode

Table B.31: P2P Advanced Mode Configurations – AI mode	
Description	Retrieves information about the Peer-to-Peer Advanced mode configuration for analog input.
URL Structure	**http://10.0.0.1/p2p_advancedai/slot_index** **http://10.0.0.1/p2p_advancedai/slot_index/ch_num** **where index = 0 : the core module** **1 ~ : the identifier of I/O extension slot** **where num = 0 ~ : the channel number**
HTTP Method	GET: Returns the representation of all of Peer-to-Peer Advanced mode configuration resource. PUT: Replace all of Peer-to-Peer Advanced mode configuration resource. PATCH: Apply partial modifications to Peer-to-Peer Advanced mode configurations resource.
GET	Multi Channel Request: GET /p2p_advancedai/slot_index Single Channel Request: GET /p2p_advancedai/slot_index/ch_num [Example]: ?Request : GET /p2p_advancedai/slot_1, ADAM-3651 for example Content-type: application/json Response: 200 OK { "Adv": [{ "Ch": 0, "En": 1, "InM ": 0, "OutM": 0, "Inv": 1, "AOH": 65535, "AOL": 452, "COS": 1, "Dev": 0, "PItv": 3, "IP": 5, "SL": 0, "DCh": 1, "BiUn": 0 },

GET (Cont.)	{ "Ch": 1, "En": 1, "InM ": 0, "OutM": 0, "Inv": 1, "AOH": 65535, "AOL": 452, "COS": 1, "Dev": 0, "PItv": 3, "IP": 5, "SL": 0, "DCh": 1, "BiUn": 0 }, { "Ch": 2, "En": 1, "InM ": 0, "OutM": 0, "Inv": 1, "AOH": 65535, "AOL": 452, "COS": 1, "Dev": 0, "PItv": 3, "IP": 5, "SL": 0, "DCh": 1, "BiUn": 0 }, { "Ch": 3, "En": 0, "InM ": 0, "OutM": 0, "Inv": 1, "AOH": 32768, "AOL": 0, "COS": 1, "Dev": 0, "PItv": 3, "IP": 0, "SL": 0, "DCh": 1, "BiUn": 0 }, { "Ch": 4, "En": 0, "InM ": 0, "OutM": 0, "Inv": 1, "AOH": 32768, "AOL": 0, "COS": 1, "Dev": 0, "PItv": 3, "IP": 0,

GET (Cont.)


```
      "SL": 0,
      "DCh": 1,
      "BiUn": 0
    },
    {
      "Ch": 5,
      "En": 0,
      "InM ": 0,
      "OutM": 0,
      "Inv": 1,
      "AOH": 32768,
      "AOL": 0,
      "COS": 1,
      "Dev": 0,
      "PItv": 3,
      "IP": 0,
      "SL": 0,
      "DCh": 1,
      "BiUn": 0
    },
    {
      "Ch": 6,
      "En": 0,
      "InM ": 0,
      "OutM": 0,
      "Inv": 1,
      "AOH": 32768,
      "AOL": 0,
      "COS": 1,
      "Dev": 0,
      "PItv": 3,
      "IP": 0,
      "SL": 0,
      "DCh": 1,
      "BiUn": 0
    },
    {
      "Ch": 7,
      "En": 0,
      "InM ": 0,
      "OutM": 0,
      "Inv": 1,
      "AOH": 32768,
      "AOL": 0,
      "COS": 1,
      "Dev": 0,
      "PItv": 3,
      "IP": 0,
      "SL": 0,
      "DCh": 1,
      "BiUn": 0
    }
  ]
}
```


- Request: **GET /p2p_advancedai/slot_1/ch_1**

Content-type: application/json
Response: 200 OK

GET (Cont.)	{
	"Ch": 1,
	"En": 1,
	"InM ": 0,
	"OutM": 0,
	"Inv": 1,
	"AOH": 32768,
	"AOL": 0,
	"COS": 1,
	"Dev": 0,
	"PItv": 3,
	"IP": 0,
	"SL": 0,
	"DCh": 1,
	"BiUn": 0
	}

PUT	Multi Channel Request: **PUT /p2p_advancedai/slot_index** Single Channel Request: **PUT /p2p_advancedai/slot_index/ch_num** [Example]: ■ Request: **PUT /p2p_advancedai/slot_2** Content-type: application/json { "Adv": [{ "Ch": 0, "En": 1, "InM ": 0, "OutM": 0, "Inv": 1, "AOH": 32768, "AOL": 0, "COS": 1, "Dev": 0, "PItv": 3, "IP": 0, "SL": 0, "DCh": 1, "BiUn": 0 }, { "Ch": 1, "En": 1, "InM ": 0, "OutM": 0, "Inv": 1, "AOH": 32768, "AOL": 0, "COS": 1, "Dev": 0, "PItv": 3, "IP": 0, "SL": 0, "DCh": 1, "BiUn": 0 },

PUT (Cont.)	{ "Ch": 2, "En": 1, "InM ": 0, "OutM": 0, "Inv": 1, "AOH": 32768, "AOL": 0, "COS": 1, "Dev": 0, "PItv": 3, "IP": 0, "SL": 0, "DCh": 1, "BiUn": 0 }, { "Ch": 3, "En": 0, "InM ": 0, "OutM": 0, "Inv": 1, "AOH": 32768, "AOL": 0, "COS": 1, "Dev": 0, "PItv": 3, "IP": 0, "SL": 0, "DCh": 1, "BiUn": 0 }, { "Ch": 4, "En": 0, "InM ": 0, "OutM": 0, "Inv": 1, "AOH": 32768, "AOL": 0, "COS": 1, "Dev": 0, "PItv": 3, "IP": 0, "SL": 0, "DCh": 1, "BiUn": 0 }, { "Ch": 5, "En": 0, "InM ": 0, "OutM": 0, "Inv": 1, "AOH": 32768, "AOL": 0, "COS": 1, "Dev": 0, "PItv": 3, "IP": 0,

PUT (Cont.)	"SL": 0,
	"DCh": 1,
	"BiUn": 0
	
	},
	{
	"Ch": 6,
	"En": 0,
	"InM ": 0,
	"OutM": 0,
	"Inv": 1,
	"AOH": 32768,
	"AOL": 0,
	"COS": 1,
	"Dev": 0,
	"PItv": 3,
	"IP": 0,
	"SL": 0,
	"DCh": 1,
	"BiUn": 0
	},
	{
	"Ch": 7,
	"En": 0,
	"InM ": 0,
	"OutM": 0,
	"Inv": 1,
	"AOH": 32768,
	"AOL": 0,
	"COS": 1,
	"Dev": 0,
	"PItv": 3,
	"IP": 0,
	"SL": 0,
	"DCh": 1,
	"BiUn": 0
	}
	]
	}
	Response: 200 OK
	
	■ Request: **PUT /p2p_advancedai/slot_2/ch_3**
	Content-type: application/json
	{
	"Ch": 3,
	"En": 0,
	"InM ": 0,
	"OutM": 0,
	"Inv": 1,
	"AOH": 32768,
	"AOL": 0,
	"COS": 1,
	"Dev": 0,
	"PItv": 3,
	"IP": 0,
	"SL": 0,
	"DCh": 1,
	"BiUn": 0
	}
	Response: 200 OK

PATCH	Multi Channel Request:
	PATCH /p2p_advancedai/slot_index
	Single Channel Request:
	PATCH /p2p_advancedai/slot_index/ch_num
	
	[Example]:
	■ Request: **PATCH /p2p_advancedai/slot_0/ch_15**
	Content-type: application/json
	{
	"En": 0,
	"COS": 1,
	"PItv": 5
	}
	
	Response: 200 OK

■ JSON array name definition:

Field	Abbreviation	Data Type
Array of P2P Advanced mode configurations	Adv	Array

■ Resource value definitions:

Field	Abbreviation	Data Type	Property	Description
Channel Number	Ch	Number	R	0, 1, …: P2P input channel number.
P2P Enabled	En	Number	RW	1 or 0: Enable or Disable P2P function
Channel input mode	InM	Number	Number	Input mode: 0 Normal 1 High alarm 2 Low alarm 3 High or Low alarm
Channel output mode	OutM	Number	RW	Output mode 0 DO mode 1 AO mode
Invert Signal	Inv	Number	RW	1 or 0: Enable or Disable DO invert function
Enable AI deviation	COS	Number	RW	1 or 0: Enable or Disable AI deviation. Or Alarm changed Input mode: COS Normal: Deviation High alarm / Low alarm / High or Low alarm: Change of alarm status
AI Deviation value	Dev	Number	RW	1 ~ 100: AI deviation value. (Unit: %)
Period Time	PItv	Number	RW	Period Time (unit: sec) : 0-86400 Default: 5 sec 0: disable
Destination IP Table Mask	IP	Number	RW	The mask of destination IP table Mask Bit: IP table index

				0	IP 0
				1	IP 1
				2	IP 2
				3	IP 3
				4	IP 4
				5	IP 5
				6	IP 6
				7	IP 7
				8	IP 8
				9	IP 9
				10	IP 10
				11	IP 11
				12	IP 12
				13	IP 13
				14	IP 14
				15	IP 15
Destination channel slot	SL	Number	RW	The slot index which the destination channel is owned	
Destination channel index	DCh	Number	RW	The index number of the destination channel. For example, 0 ~ 7: Do channels for ADAM-3600-A1F 0 ~ 3: Do channels for WISE-4060	
Enable AI Bi-polar to Uni-polar conversion [Only positive value valid]	BiUn	Number	RW	1 or 0: Enable or Disable AI Bi-polar to Uni-polar conversion for Slot 0 (Core module).	
AO output value for DI high or AI Alarm	AOH	Number	RW	0-65535: AO output value for DI status is high or AI alarm occur	
AO output value for DI low or no AI Alarm	AOL	Number	RW	0-65535: AO output value for DI status is low or no AI alarm occur	
Remarks					

B.2.30 P2P Advanced Mode Configurations – Sensor mode

Table B.32: P2P Advanced Mode Configurations – Sensor mode	
Description	Retrieves information about the Peer-to-Peer Advanced mode configuration for analog input.
URL Structure	**http://10.0.0.1/p2p_advancedsen/slot_index** **http://10.0.0.1/p2p_advancedsen/slot_index/ch_num** **where index = 0 : the core module** **1 ~ : the identifier of I/O extension slot** **where num = 0 ~ : the channel number**
HTTP Method	GET: Returns the representation of all of Peer-to-Peer Advanced mode configuration resource. PUT: Replace all of Peer-to-Peer Advanced mode configuration resource. PATCH: Apply partial modifications to Peer-to-Peer Advanced mode configurations resource.
GET	Multi Channel Request: **GET /p2p_advancedsen/slot_index** Single Channel Request: **GET /p2p_advancedsen/slot_index/ch_num**

GET (Cont.)	[Example]: ■ Request: **GET /p2p_advancedsen/slot_0** Content-type: application/json Response: 200 OK { "Adv": [{ "Ch": 0, "En": 1, "InM ": 0, "OutM": 0, "Inv": 1, "AOH": 65535, "AOL": 452, "COS": 1, "Dev": 0, "PItv": 3, "IP": 5, "SL": 0, "DCh": 1, "BiUn": 0 }, { "Ch": 1, "En": 1, "InM ": 0, "OutM": 0, "Inv": 1, "AOH": 65535, "AOL": 452, "COS": 1, "Dev": 0, "PItv": 3, "IP": 5, "SL": 0, "DCh": 1, "BiUn": 0 }, { "Ch": 2, "En": 1, "InM ": 0, "OutM": 0, "Inv": 1, "AOH": 65535, "AOL": 452, "COS": 1, "Dev": 0, "PItv": 3, "IP": 5, "SL": 0, "DCh": 1, "BiUn": 0 }, { "Ch": 3, "En": 0, "InM ": 0, "OutM": 0,

GET (Cont.)	"Inv": 1, "AOH": 32768, "AOL": 0, "COS": 1, "Dev": 0, "PItv": 3, "IP": 0, "SL": 0, "DCh": 1, "BiUn": 0 }, { "Ch": 4, "En": 0, "InM ": 0, "OutM": 0, "Inv": 1, "AOH": 32768, "AOL": 0, "COS": 1, "Dev": 0, "PItv": 3, "IP": 0, "SL": 0, "DCh": 1, "BiUn": 0 }, { "Ch": 5, "En": 0, "InM ": 0, "OutM": 0, "Inv": 1, "AOH": 32768, "AOL": 0, "COS": 1, "Dev": 0, "PItv": 3, "IP": 0, "SL": 0, "DCh": 1, "BiUn": 0 }, { "Ch": 6, "En": 0, "InM ": 0, "OutM": 0, "Inv": 1, "AOH": 32768, "AOL": 0, "COS": 1, "Dev": 0, "PItv": 3, "IP": 0, "SL": 0, "DCh": 1, "BiUn": 0 }, {

GET (Cont.)	"Ch": 7,
	"En": 0,
	"InM ": 0,
	"OutM": 0,
	"Inv": 1,
	"AOH": 32768,
	"AOL": 0,
	"COS": 1,
	"Dev": 0,
	"PItv": 3,
	"IP": 0,
	"SL": 0,
	"DCh": 1,
	"BiUn": 0
	}
	]
	}
	
	■ Request: **GET /p2p_advancedsen/slot_1/ch_1**
	
	Content-type: application/json
	Response: 200 OK
	{
	"Ch": 1,
	"En": 1,
	"InM ": 0,
	"OutM": 0,
	"Inv": 1,
	"AOH": 32768,
	"AOL": 0,
	"COS": 1,
	"Dev": 0,
	"PItv": 3,
	"IP": 0,
	"SL": 0,
	"DCh": 1,
	"BiUn": 0
	}

PUT	Multi Channel Request: **PUT /p2p_advancedsen/slot_index** Single Channel Request: **PUT /p2p_advancedsen/slot_index/ch_num** [Example]: ■ Request: **PUT /p2p_advancedsen/slot_2** Content-type: application/json { "Adv": [{ "Ch": 0, "En": 1, "InM ": 0, "OutM": 0, "Inv": 1, "AOH": 32768, "AOL": 0, "COS": 1, "Dev": 0,

PUT (Cont.)	"PItv": 3, "IP": 0, "SL": 0, "DCh": 1, "BiUn": 0 }, { "Ch": 1, "En": 1, "InM ": 0, "OutM": 0, "Inv": 1, "AOH": 32768, "AOL": 0, "COS": 1, "Dev": 0, "PItv": 3, "IP": 0, "SL": 0, "DCh": 1, "BiUn": 0 }, { "Ch": 2, "En": 1, "InM ": 0, "OutM": 0, "Inv": 1, "AOH": 32768, "AOL": 0, "COS": 1, "Dev": 0, "PItv": 3, "IP": 0, "SL": 0, "DCh": 1, "BiUn": 0 }, { "Ch": 3, "En": 0, "InM ": 0, "OutM": 0, "Inv": 1, "AOH": 32768, "AOL": 0, "COS": 1, "Dev": 0, "PItv": 3, "IP": 0, "SL": 0, "DCh": 1, "BiUn": 0 }, { "Ch": 4, "En": 0, "InM ": 0, "OutM": 0, "Inv": 1,

PUT (Cont.)	"AOH": 32768,
	"AOL": 0,
	"COS": 1,
	"Dev": 0,
	"PItv": 3,
	"IP": 0,
	"SL": 0,
	"DCh": 1,
	"BiUn": 0
	},
	{
	"Ch": 5,
	"En": 0,
	"InM ": 0,
	"OutM": 0,
	"Inv": 1,
	"AOH": 32768,
	"AOL": 0,
	"COS": 1,
	"Dev": 0,
	"PItv": 3,
	"IP": 0,
	"SL": 0,
	"DCh": 1,
	"BiUn": 0
	
	},
	{
	"Ch": 6,
	"En": 0,
	"InM ": 0,
	"OutM": 0,
	"Inv": 1,
	"AOH": 32768,
	"AOL": 0,
	"COS": 1,
	"Dev": 0,
	"PItv": 3,
	"IP": 0,
	"SL": 0,
	"DCh": 1,
	"BiUn": 0
	},
	{
	"Ch": 7,
	"En": 0,
	"InM ": 0,
	"OutM": 0,
	"Inv": 1,
	"AOH": 32768,
	"AOL": 0,
	"COS": 1,
	"Dev": 0,
	"PItv": 3,
	"IP": 0,
	"SL": 0,
	"DCh": 1,
	"BiUn": 0
	}
	]

PUT (Cont.)

```
}
Response: 200 OK

■ Request: PUT /p2p_advancedsen/slot_2/ch_3
Content-type: application/json
{
    "Ch": 3,
    "En": 0,
    "InM ": 0,
    "OutM": 0,
    "Inv": 1,
    "AOH": 32768,
    "AOL": 0,
    "COS": 1,
    "Dev": 0,
    "PItv": 3,
    "IP": 0,
    "SL": 0,
    "DCh": 1,
    "BiUn": 0
}
Response: 200 OK
```

PATCH

Multi Channel Request:
PATCH /p2p_advancedsen/slot_index
Single Channel Request:
PATCH /p2p_advancedsen/slot_index/ch_num

[Example]:

■ Request: **PATCH /p2p_advancedsen/slot_0/ch_15**

```
Content-type: application/json
{
    "En": 0,
    "COS": 1,
    "PItv": 5
}

Response: 200 OK
```

■ JSON array name definition:

Field	Abbreviation	Data Type
Array of P2P Advanced mode configurations	Adv	Array

■ Resource value definitions:

Field	Abbreviation	Data Type	Property	Description
Channel Number	Ch	Number	R	0, 1, …: P2P input channel number.
P2P Enabled	En	Number	RW	1 or 0: Enable or Disable P2P function
Channel input mode	InM	Number	RW	Input mode: 0 High alarm 1 Low alarm 2 High or Low alarm
Channel output mode	OutM	Number	RW	Output mode 0 DO mode 1 AO mode

Invert Signal	Inv	Number	RW	1 or 0: Enable or Disable DO invert function
Enable AI deviation	COS	Number	RW	1 or 0: Enable or Disable AI deviation. Or Alarm changed

Input mode	COS
High alarm	Change of alarm status
Low alarm	
High or Low alarm	

Period Time	PItv	Number	RW	Period Time (unit: sec) : 0-86400 Default: 5 sec 0: disable
Destination IP Table Mask	IP	Number	RW	The mask of destination IP table

Mask Bit	IP table index
0	IP 0
1	IP 1
2	IP 2
3	IP 3
4	IP 4
5	IP 5
6	IP 6
7	IP 7
8	IP 8
9	IP 9
10	IP 10
11	IP 11
12	IP 12
13	IP 13
14	IP 14
15	IP 15

Destination channel slot	SL	Number	RW	The slot index which the destination channel is owned
Destination channel index	DCh	Number	RW	The index number of the destination channel. For example, 0 ~ 7: Do channels for ADAM-3600-A1F 0 ~ 3: Do channels for WISE-4060
AO output value for DI high or AI Alarm	AOH	Number	RW	0-65535: AO output value for DI status is high or AI alarm occur
AO output value for DI low or no AI Alarm	AOL	Number	RW	0-65535: AO output value for DI status is low or no AI alarm occur
Remarks				

B.2.31 P2P Configuration file

Table B.33: P2P Configuration file	
Description	Retrieves the JSON file about the all Peer-to-Peer configuration.
URL Structure	**http://10.0.0.1/p2p_config.json**
HTTP Method	GET: Returns the file of all of Peer-to-Peer configuration resource. PUT: None PATCH: None

GET	Request:
	GET /p2p_config.json
	
	[Example]:
	■ Request: **GET /p2p_ config.json**
	
	Content-type: application/json
	Response: 200 OK
	{
	"Id": "WISE-4012",
	"p2p_mode": {
	"Md": 0,
	"En": 1,
	"QL": 0,
	"ECS": 1,
	"PpG": 5048
	},
	"p2p_iptable":{
	"IpTbl": [
	{
	"Idx": 0,
	"IP": "172.18.3.5",
	"Id": 27488,
	"Pw ":"0F0F0F0F0F0F0F0F"
	},
	{
	"Idx": 1,
	"IP": "172.18.3.5",
	"Id": 27472,
	"Pw ":"0F0F0F0F0F0F0F0F"
	},
	{
	"Idx": 2,
	"IP": "255.255.255.255",
	"Id": 27488,
	"Pw ":"0F0F0F0F0F0F0F0F"
	},
	{
	"Idx": 3,
	"IP": "255.255.255.255",
	"Id": 27488,
	"Pw ":"0F0F0F0F0F0F0F0F"
	},
	{
	"Idx": 4,
	"IP": "255.255.255.255",
	"Id": 27488,
	"Pw ":"0F0F0F0F0F0F0F0F"
	},

GET (Cont.)	{
	"Idx": 5,
	"IP": "255.255.255.255",
	"Id": 27488,
	"Pw ":"0F0F0F0F0F0F0F0F"
	},
	{
	"Idx": 6,
	"IP": "255.255.255.255",
	"Id": 27488,
	"Pw ":"0F0F0F0F0F0F0F0F"
	},
	……
	……
	{
	"Idx": 16,
	"IP": "255.255.255.255",
	"Id": 27488,
	"Pw ":"0F0F0F0F0F0F0F0F"
	}
	]
	},
	"p2p_basic":{
	"DIChM ": [15,0,0,0,0],
	"AIChM": [0,0,15,0,0],
	"Inv": [15,0,0,0,0],
	"COS": 1,
	"Dev": 5,
	"PItv": 3,
	"IP": "172.18.3.55",
	"Id": 27488,
	"BiUn": [0, 0, 15, 0, 0]
	},
	"p2p_advanceddi/slot_0":{
	"Adv": [
	{
	"Ch": 0,
	"En": 1,
	"OutM": 0,
	"Inv": 1,
	"AOH": 65535,
	"AOL": 452,
	"COS": 1,
	"PItv": 3,
	"IP": 8,
	"SL": 0,
	"DCh": 1
	},

GET (Cont.)	{
	"Ch": 1,
	"En": 1,
	"OutM": 0,
	"Inv": 1,
	"AOH": 32768,
	"AOL": 0,
	"COS": 1,
	"Pltv": 3,
	"IP": 4,
	"SL": 0,
	"DCh": 0
	},
	{
	"Ch": 2,
	"En": 1,
	"OutM": 0,
	"Inv": 1,
	"AOH": 32768,
	"AOL": 0,
	"COS": 1,
	"Pltv": 3,
	"IP": 5,
	"SL": 0,
	"DCh": 5
	}
	]
	},
	"p2p_advancedai/slot_0":{
	"Adv": [
	{
	"Ch": 0,
	"En": 1,
	"InM ": 0,
	"OutM": 0,
	"Inv": 1,
	"AOH": 65535,
	"AOL": 452,
	"COS": 1,
	"Dev": 0,
	"Pltv": 3,
	"IP": 5,
	"SL": 0,
	"DCh": 1,
	"BiUn": 0
	},

GET (Cont.)


```
    {
      "Ch": 1,
      "En": 1,
      "InM ": 0,
      "OutM": 0,
      "Inv": 1,
      "AOH": 65535,
      "AOL": 452,
      "COS": 1,
      "Dev": 0,
      "PItv": 3,
      "IP": 5,
      "SL": 0,
      "DCh": 1,
      "BiUn": 0
    },
    {
      "Ch": 2,
      "En": 1,
      "InM ": 0,
      "OutM": 0,
      "Inv": 1,
      "AOH": 65535,
      "AOL": 452,
      "COS": 1,
      "Dev": 0,
      "PItv": 3,
      "IP": 5,
      "SL": 0,
      "DCh": 1,
      "BiUn": 0
    },
    {
      "Ch": 3,
      "En": 0,
      "InM ": 0,
      "OutM": 0,
      "Inv": 1,
      "AOH": 32768,
      "AOL": 0,
      "COS": 1,
      "Dev": 0,
      "PItv": 3,
      "IP": 0,
      "SL": 0,
      "DCh": 1,
      "BiUn": 0
    }
  ]
 }
}
```


■ Resource value definitions:

Field	Abbreviation	Data Type	Property	Description
Module name	Id	String	R	The module name of the configuration file
P2P mode and command configuration	p2p_mode	Object	R	Reference Mode & Common Configurations

P2P destination IP table	p2p_iptable	Object	R	Reference Destination IP Table
P2P basic configuration	p2p_basic	Object	R	Reference Basic Mode Configurations
P2P DI advance configuration	p2p_advanceddi	Object	R	Reference Advanced Mode Configurations – DI mode
P2P AI advance configuration	p2p_advancedai	Object	R	Reference Advanced Mode Configurations – AI mode
P2P sensor advance configuration	p2p_advancedsen	Object	R	Reference Advanced Mode Configurations – sensor mode

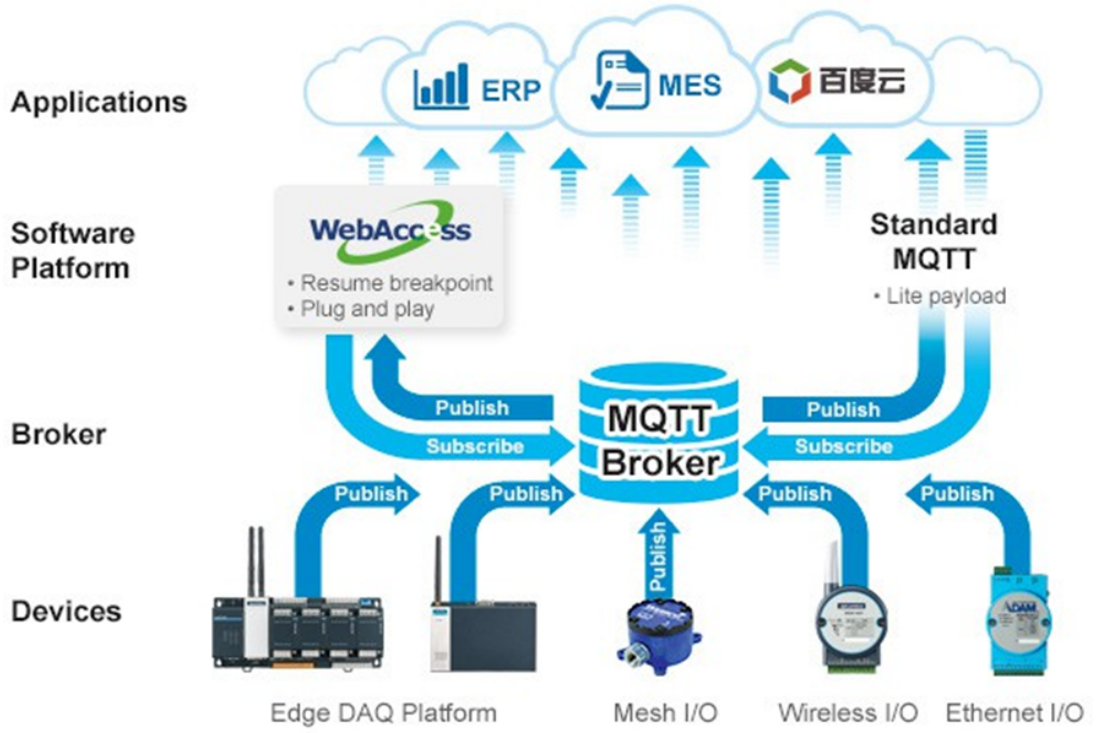
Table B.34: WISE-4250 supported Ciphers List		
Module	**Security**	**Ciphers**
WISE-4250	TLS 1.2 ~ 1.3	TLS_AES_128_GCM_SHA256 TLS_AES_128_CCM_SHA256 TLS_AES_128_CCM_8_SHA256 TLS_ECDHE_ECDSA_WITH_AES_256_CCM TLS_DHE_RSA_WITH_AES_256_CCM TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA TLS_DHE_RSA_WITH_AES_256_CBC_SHA TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8 TLS_DHE_RSA_WITH_AES_256_CCM_8 TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA256 TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_ECDSA_WITH_AES_128_CCM TLS_DHE_RSA_WITH_AES_128_CCM TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA TLS_DHE_RSA_WITH_AES_128_CBC_SHA TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 TLS_DHE_RSA_WITH_AES_128_CCM_8 TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_GCM_SHA256 TLS_ECDHE_RSA_WITH_CAMELLIA_128_GCM_SHA256 TLS_DHE_RSA_WITH_CAMELLIA_128_GCM_SHA256 TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_CBC_SHA256 TLS_ECDHE_RSA_WITH_CAMELLIA_128_CBC_SHA256 TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA256 TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA TLS_ECDHE_ECDSA_WITH_ARIA_128_GCM_SHA256 TLS_ECDHE_RSA_WITH_ARIA_128_GCM_SHA256 TLS_DHE_RSA_WITH_ARIA_128_GCM_SHA256 TLS_ECDHE_ECDSA_WITH_ARIA_128_CBC_SHA256 TLS_ECDHE_RSA_WITH_ARIA_128_CBC_SHA256 TLS_DHE_RSA_WITH_ARIA_128_CBC_SHA256 TLS_RSA_WITH_AES_256_CCM TLS_RSA_WITH_AES_256_CBC_SHA256 TLS_RSA_WITH_AES_256_CBC_SHA TLS_ECDH_RSA_WITH_AES_256_CBC_SHA TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA TLS_RSA_WITH_AES_256_CCM_8 TLS_RSA_WITH_CAMELLIA_256_CBC_SHA256 TLS_RSA_WITH_CAMELLIA_256_CBC_SHA TLS_RSA_WITH_AES_128_GCM_SHA256 TLS_RSA_WITH_AES_128_CCM TLS_RSA_WITH_AES_128_CBC_SHA256 TLS_RSA_WITH_AES_128_CBC_SHA TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256 TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256 TLS_ECDH_RSA_WITH_AES_128_CBC_SHA TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256 TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256

Table B.34: WISE-4250 supported Ciphers List
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_128_CCM_8
TLS_RSA_WITH_CAMELLIA_128_GCM_SHA256
TLS_RSA_WITH_CAMELLIA_128_CBC_SHA256
TLS_RSA_WITH_CAMELLIA_128_CBC_SHA
TLS_ECDH_RSA_WITH_CAMELLIA_128_GCM_SHA256
TLS_ECDH_RSA_WITH_CAMELLIA_128_CBC_SHA256
TLS_ECDH_ECDSA_WITH_CAMELLIA_128_GCM_SHA256
TLS_ECDH_ECDSA_WITH_CAMELLIA_128_CBC_SHA256
TLS_ECDH_ECDSA_WITH_ARIA_128_GCM_SHA256
TLS_ECDH_RSA_WITH_ARIA_128_GCM_SHA256
TLS_RSA_WITH_ARIA_128_GCM_SHA256
TLS_ECDH_ECDSA_WITH_ARIA_128_CBC_SHA256
TLS_ECDH_RSA_WITH_ARIA_128_CBC_SHA256
TLS_RSA_WITH_ARIA_128_CBC_SHA256

Appendix C

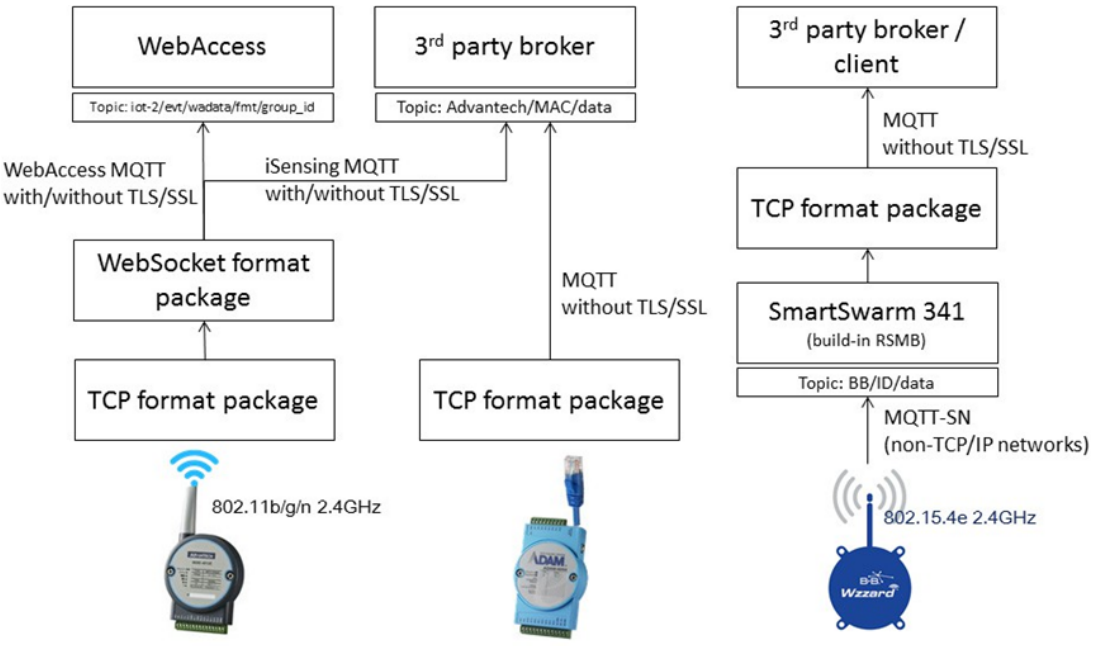
iSensing MQTT

MQTT (message queuing telemetry transport) is a publish/subscribe messaging protocol for constrained Internet of Things (IoT) devices in low-bandwidth, high-latency, or unreliable networks. Advantech iSensing MQTT is an Advantech-defined MQTT topic and payload for iSensing and iConnectivity devices, including WISE-4000 wireless I/O, ADAM-6000 Ethernet I/O, and B+B Wizard mesh I/O.



There are two differences between WISE-4000 wireless I/O and ADAM-6000 Ethernet I/O:

1. WISE supports TLS/SSL but ADAM does not.
2. WISE uses a web socket format package and ADAM uses TCP format package.



C.1 iSensing MQTT Format for publishing

C.1.1 Topic Format: Advantech/IMEI/data

Table C.1: Topic Format: Advantech/IMEI/data

Field	Description
Advantech	Vendor of WISE Series
MAC (IMEI)	MAC address or IMEI of WISE Series
data	Topic for query the data of WISE Series

C.1.2 Payload Format

Table C.2: Payload Format

Field	Abbreviation	Description
Sequence Number	s	Sequence number that updates each time a sensor reading is made 0~9, 0~9, …
Time Stamp	t	ISO 8601 timestamp of the UTC time when the sensor reading was made
Quality	q	Quality of the sensor reading 192: The value is good. There are no special conditions.
Configuration Change	c	Configuration change affecting the reading (e.g., scaling, calibration) 0~9, 0~9, …

Table C.2: Payload Format		
Digital Input	di	Digital Input Mode true/false Counter 0~4294967295 Frequency Mode 0~30000 (default unit: 0.1Hz), 0~300000 (default unit: 0.01Hz)
Digital Output	do	Digital Output: true/false
Analog Input	ai	Engineering value of analog input: -10000.000~10000.000 (unit: mV) -20.000~20.000 (unit: mA)
Temperature Sensor	temp	Temperature engineering data, the value is floating type. Unit: According range code For example, 999.120 -> 999.12 °C -3.220 -> -3.22 °C
Temperature Range Code	temprc	4096: Temperature (°C) 4097: Temperature (°F) 4098: Temperature (K)
Humidity Sensor	rh	Humidity engineering data, the value is floating type. Unit: According range code
Humidity Range Code	rhrc	4128: Humidity (%)
RS-485	p1v06r0012x05	p1: Port number v01: Server ID s: Coil (r: Register) 00b1: Modbus/RTU address x02: Mapping channel

C.1.3 Example

Table C.3: Example		
Model Name	**JSON Data**	**Notes**
WISE-4250-S232 (1-Temperature, 1-Humidity)	{"s":6, "t":"2017-11-03T15:06:16Z", "q":192, "c":2, "temp1":23.6, "temp1rc":4096, "rh1":43.6, "rh1rc":4128 }	■ Temperature and range code in the configured engineering units ■ The humidity in % unit and range code.
WISE-4250-S214 (4-AI, 4-DI)	{"s":1, "t":"2014-07-11T15:26:37Z", "q":192,"c":1, "ai1":-1.234567, "ai_st1":1, "ai2":-123.4567, "ai_st1":2 "di1":false,"di2":true }	The value of the sensor in engineering units if ai disable the value is 9999.9999 "ai_st": AI status number meaning: 0, AI Channel disable 1, Streaming 2, High latch 3, High momentary 4, Low latch 5, Low momentary The value of di1 and di2 can be: false, true.

Table C.3: Example		
WISE-4250-S250 (6-DI, 2-DO, 1-RS-485)	{"s":1, "t":"2014-07-11T15:26:37Z", "q":192,"c":1, "di1":false,"di2":true "do1":false,"do2":true "p1vf-fr0001x01":348, "p1v06r0012x05":32768 "p1v01s00b1x03":true, "p1v12s012dx0f":false}	■ The value of di1 and di2 can be: false, true. ■ The value of do1 and do2 can be: false, true ■ The value of coil can be: true, false. ■ The value of register can be 0x0 to 0xFFFF.
WISE-4250-S251 (6-DI, 1-RS-485)	{"s":3, "t":"2024-05-10T06:13:08Z", "q":192,"c":4, "di1":false,"di2":false,"di3":false," di4":false,"di5":false,"di6":false, "p1v01s0011x00":false, "p1v01s0012x01":true, "p1v01s0013x02":false, "p1v01s0014x03":true, "p1v01s0015x04":true, "p1v01s0016x05":false, "p1v01s0017x06":false, "p1v01s0018x07":false}	■ The value of di1~di6 can be: false, true. ■ The value of coil can be: true, false. ■ The value of register can be 0x0 to 0xFFFF.

Note! *The channel number on WISE module is "0" based, and the channel number of MQTT topic is "1" based. For example: "DO0" of WISE module use topic "do1".*

C.2 iSensing MQTT Format for subscribing

C.2.1 Topic Format: Advantech/IMEI/ctl/doIndex

These topics are used to control the digital outputs on the sensors that support them. These requests need to be published to the broker handling the Sensor network.

Table C.4: Topic Format: Advantech/IMEI/ctl/doIndex	
Field	**Description**
Advantech	Vendor of WISE Series
IMEI	MAC address of WISE Series
doIndex	The index of the DO channel. Note that the index start with '1'

Note! *The channel number on WISE module is "0" based, and the channel number of MQTT topic is "1" based. For example: "DO0" of WISE module use topic "do1".*

C.2.2 Payload Format

Table C.5: Payload Format	
JSON data	**Description**
{"v":true}	Setup the DO Boolean value as true
{"v":false}	Setup the DO Boolean value as false

Note! *While sending control MQTT command, do not set the retain bit when publishing messages to this topic. Otherwise, an old retained message may change the state of the output.*

Note: The command rule of controlling WISE-4250-S250's DO is described below.

First, please note that, in the payload of "Advantech/MAC of WISE/data" topic, the "do2" represent DO channel 1 of WISE-4250-S250. The rule of naming is shown as below.

do(Number+1) => DO channel number of WISE-4250-S250.

Second, MQTT downlink's rule is shown as following.

Topic Rule: Advantech/MAC of WISE/ctl/key of certain DO channel

Payload Rule: {"v": Value(Boolean)}

Topic Example: Advantech/74FE4864CC75/ctl/do2

Payload Example: {"v": true}

If you'd like to learn more details, please refer to the FAQ content or send an email to inquire with Advantech's technical staff.

FAQ: How to use MQTT downlink to control WISE-4000's DO or Modbus RTU Server?

Appendix D

REST Resource

D.1 Introduction

REpresentational State Transfer (REST) is a design style of software architecture for Web application behaves and services including image indication, resource request and response and message delivery. It can be developed compatible with popular protocols or standards like HTTP, URI, JSON, HTML. With the advantage of scalability, simplicity and performance, it's already adopted in Web service by Amazon, Yahoo. The Web service of is developed based on HTML5 language, if user need to integrate this into other Web services, the following information/command list should be referred for implementation.

D.2 REST Resources

D.2.1 Digital Input

D.2.1.1 /di_value/slot_index/ch_num

Table D.1: /di_value/slot_index/ch_num	
Description	Retrieves information about the digital input value resource on specific slot.
URL Structure	http://10.0.0.1/di_value/slot_index http://10.0.0.1/di_value/slot_index/ch_num
HTTP Method	GET:Returns the representation of all of digital input value resource. PUT:Replace all of digital input value resource PATCH:Apply partial modifications to digital input value resource.

GET	Multiple Channel Request:
	GET /di_value/slot_index
	Single Channel Request:
	GET /di_value/slot_index/ch_num
	
	[Example]
	
	Request: **GET /di_value/slot_0**
	
	Content-type: application/json
	Response: 200 OK
	{
	"DIVal": [
	{
	"Ch":0,
	"Md":0,
	"Stat":1,
	"Val":1,
	"Cnting":0,
	"ClrCnt":0,
	"OvLch": 0
	},
	{
	"Ch":1,
	"Md":0,
	"Stat":0,
	"Val":0,
	"Cnting":0,
	"ClrCnt":0,
	"OvLch": 0
	},
	{
	"Ch":2,
	"Md":1,
	"Stat":0,
	"Val":3378,
	"Cnting":1,
	"ClrCnt":0,
	"OvLch": 0
	},
	{
	"Ch":3,
	"Md":3,
	"Stat":0,
	"Val":1,
	"Cnting":0,
	"ClrCnt":0,
	"OvLch": 0
	}
	]
	}
	
	Request : **GET /di_value/slot_0/ch_2**
	
	Content-type: application/json
	Response: 200 OK
	{
	"Ch":2,
	"Md":0,
	"Stat":1,
	"Val":1,
	"Cnting":0,
	"ClrCnt":0,
	"OvLch": 0
	}

PUT	Single/Multiple Channel Request:
	PUT /di_value/slot_index
	Single Channel Request:
	PUT /di_value/slot_index/ch_num
	
	[Example]
	
	Request: **PUT /di_value/slot_0**
	
	Content-type: application/json
	{
	"DIVal": [
	
	{
	"Ch":0,
	"Md":0,
	"Stat":0,
	"Val":0,
	"Cnting":0,
	"ClrCnt":0,
	"OvLch": 0
	},
	{
	"Ch":1,
	"Md":0,
	"Stat":0,
	"Val":0,
	"Cnting":0,
	"ClrCnt":0,
	"OvLch": 0
	},
	{
	"Ch":2,
	"Md":1,
	"Stat":0,
	"Val":3378,
	"Cnting":0,
	"ClrCnt":1,
	"OvLch": 0
	},
	{
	"Ch":3,
	"Md":3,
	"Stat":0,
	"Val":0,
	"Cnting":0,
	"ClrCnt":0,
	"OvLch": 0
	}
	]
	}
	
	Response: 200 OK
	
	Request: **PUT /di_value/slot_0/ch_2**
	
	Content-type: application/json
	{
	"Ch":2,
	"Md":1,
	"Stat":0,
	"Val":3378,
	"Cnting":0,
	"ClrCnt":1,
	"OvLch": 0
	}
	
	Response: 200 OK

PATCH	Single/Multiple Channel Request:
	PATCH /di_value/slot_index
	Single Channel Request:
	PATCH /di_value/slot_index/ch_num
	
	[Example]
	
	Request: **PATCH /di_value/slot_0**
	
	Content-type: application/json
	{
	"DIVal": [
	{
	"Ch":2,
	"Cnting": 1
	},
	{
	"Ch":3,
	"OvLch":0
	}
	]
	}
	
	Response: 200 OK
	
	Request: **PATCH /di_value/slot_0/ch_3**
	Content-type: application/json
	{
	"Ch":3,
	"ClrCnt":1
	}
	
	Response: 200 OK

- JSON array name definition:

Field	Abbreviation	Data Type
Array of Digital input configurations	DIVal	Array

- Resource value definitions:

<table>
<tr><th>Field</th><th>Abbreviation</th><th>Data Type</th><th>Property</th><th>Description</th></tr>
<tr><td>Channel Number</td><td>Ch</td><td>Number</td><td>R</td><td>0, 1, …: Digital input channel number.</td></tr>
<tr><td>Mode</td><td>Md</td><td>Number</td><td>R</td><td>Digital input mode.
<table><tr><td>0</td><td>DI</td></tr><tr><td>1</td><td>Counter</td></tr><tr><td>2</td><td>LowToHighLatch</td></tr><tr><td>3</td><td>HighToLowLatch</td></tr><tr><td>4</td><td>Frequency</td></tr></table></td></tr>
<tr><td>Signal Logic Status</td><td>Stat</td><td>Number</td><td>R</td><td>1, 0: Input signal is Logic High or Low.</td></tr>
<tr><td>Channel Value</td><td>Val</td><td>Number</td><td>R</td><td>DI measurement data
<table><tr><td>Input Mode</td><td>Value Description</td></tr><tr><td>DI</td><td>Logic Status of DI</td></tr><tr><td>Counter</td><td>Counter Value</td></tr><tr><td>LowToHighLatch</td><td>Logic status of DI</td></tr><tr><td>HighToLowLatch</td><td>Logic status of DI</td></tr><tr><td>Frequency</td><td>Frequency(unity 0.1 Hz)</td></tr></table></td></tr>
<tr><td>Start Counter</td><td>Cnting</td><td>Number</td><td>RW</td><td>Start/Stop counter counting
Read
1 : counter is counting
0 : not counting
Write
1 : start counting
0 : stop counting</td></tr>
<tr><td>Clear Counter</td><td>ClrCnt</td><td>Number</td><td>W</td><td>1 : Clear the counter value</td></tr>
<tr><td>Get/Clear Counter Overflow or Latch Status</td><td>OvLch</td><td>Number</td><td>RW</td><td>counter overflow or latch status
Read
1 : overflow/latch occurred.
0 : no overflow or latch
Write
0 : clear the overflow or latch status</td></tr>
</table>

D.2.2 Digital Output

D.2.2.1 /do_value/slot_index/ch_num

Table D.2: /do_value/slot_index/ch_num	
Description	Retrieves information about the digital output value resource on specific slot.
URL Structure	http://10.0.0.1/do_value/slot_index http://10.0.0.1/do_value/slot_index/ch_num
HTTP Method	GET:Returns the representation of all of digital output value resource. PUT:Replace all of digital output value resource PATCH:Apply partial modifications to digital output value resource.

GET	Multiple Channel Request:
	GET /do_value/slot_index
	Single Channel Request:
	GET /do_value/slot_index/ch_num
	
	[Example]
	
	Request: **GET /do_value/slot_0**
	
	Content-type: application/json
	Response: 200 OK
	{
	"DOVal": [
	{
	"Ch":0,
	"Md":0,
	"Stat":1,
	"Val":1,
	"PsCtn":0,
	"PsStop":0,
	"PsIV": 0
	},
	{
	"Ch":1,
	"Md":0,
	"Stat":0,
	"Val":0,
	"PsCtn":0,
	"PsStop":0,
	"PsIV": 0
	},
	{
	"Ch":2,
	"Md":1,
	"Stat":1,
	"Val":3378,
	"PsCtn":0,
	"PsStop":0,
	"PsIV": 0
	},
	{
	"Ch":3,
	"Md":3,
	"Stat":1,
	"Val":1,
	"PsCtn":0,
	"PsStop":0,
	"PsIV": 0
	}
	]
	}
	
	Request : **GET /do_value/slot_0/ch_2**
	
	Content-type: application/json
	Response: 200 OK
	{
	"Ch":2,
	"Md":0,
	"Stat":1,
	"Val":1,
	"PsCtn":0,
	"PsStop":0,
	"PsIV": 0
	}

PUT	Single/Multiple Channel Request:
	PUT /do_value/slot_index
	Single Channel Request:
	PUT /do_value/slot_index/ch_num
	
	[Example]
	
	Request: **PUT /do_value/slot_0**
	
	Content-type: application/json
	{
	"DOVal": [
	{
	"Ch":0,
	"Md":0,
	"Stat":1,
	"Val":1,
	"PsCtn":0,
	"PsStop":0,
	"PsIV": 0
	},
	{
	"Ch":1,
	"Md":0,
	"Stat":0,
	"Val":0,
	"PsCtn":0,
	"PsStop":0,
	"PsIV": 0
	},
	{
	"Ch":2,
	"Md":1,
	"Stat":1,
	"Val":3378,
	"PsCtn":0,
	"PsStop":0,
	"PsIV": 0
	},
	{
	"Ch":3,
	"Md":3,
	"Stat":1,
	"Val":1,
	"PsCtn":0,
	"PsStop":0,
	"PsIV": 0
	}
	]
	}
	
	Response: 200 OK
	
	Request: **PUT /do_value/slot_0/ch_2**
	
	Content-type: application/json
	{
	"Ch":2,
	"Md":2,
	"Stat":0,
	"Val":0,
	"PsCtn":0,
	"PsStop":0,
	"PsIV": 0
	}
	
	Response: 200 OK

PATCH	Single/Multiple Channel Request:
	PATCH /do_value/slot_index
	Single Channel Request:
	PATCH /do_value/slot_index/ch_num
	
	[Example]
	
	Request: **PATCH /do_value/slot_0**
	
	Content-type: application/json
	{
	"DOVal": [
	{
	"Ch":2,
	"Md": 2
	},
	{
	"Ch":3,
	"PsStop":1
	}
	]
	}
	
	Response: 200 OK
	
	Request: **PATCH /do_value/slot_0/ch_3**
	
	Content-type: application/json
	{
	"Ch":3,
	"PsCtn":1
	}
	
	Response: 200 OK

- JSON array name definition:

Field	Abbreviation	Data Type
Array of Digital input configurations	DOVal	Array

■ Resource value definitions:

Field	Abbreviation	Data Type	Property	Description
Channel Number	Ch	Number	R	0, 1, …: Digital output channel number.
Mode	Md	Number	R	Digital output mode. 0 DO 1 Pulse Output 2 LowToHighDelay 3 HighToLowDelay
Signal Logic Status	Stat	Number	R	1, 0: Output signal is Logic High or Low.
Channel Value	Val	Number	RW	DO measurement data Output Mode Value Description DO Get the current signal status or set its status Pulse Output Get or set the absolute pulse count value LowToHighDelay Get the current signal status or set its status HighToLowDelay Get the current signal status or set its status
Pulse Output Continue State	PsCtn	Number	RW	1 / 0: Pulse outputting is continuous or not.
Stop Pulse Output	PsStop	Number	W	1: Stop the pulse outputting. (Continue is disabled, Absolute and incremental values are reset to zero. DO signal status is set to logic low.)
Incremental Pulse Output Value	PsIV	Number	RW	Incremental Pulse Output Value

D.2.3 Analog Input

D.2.3.1 /ai_value/slot_index/ch_num

Table D.3: /do_value/slot_index/ch_num	
Description	Retrieves information about the analog input value resource on specific slot.
URL Structure	http://10.0.0.1/ai_value/slot_index http://10.0.0.1/ai_value/slot_index/ch_num
HTTP Method	GET:Returns the representation of all of analog input value resource. PUT:None PATCH:Apply partial modifications to analog input value resource.

GET	Multiple Channel Request: **GET /ai_value/slot_index** Single Channel Request: GET /ai_value/slot_index/ch_num [Example] Request : **GET /ai_value/slot_0** Content-type: application/json Response: 200 OK { "AIVal": [{ "Ch":0, "En":1, "Rng":328, "Val":148, "Eg":650, "Evt":0, "LoA": 0, "HiA": 0, "HVal":190, "HEg":1250, "LVal":15, "LEg":500, "SVal":148, "ClrH": 0, "ClrL": 0 }, { "Ch":1, "En":1, "Rng":328, "Val":0, "Eg":0, "Evt":0, "LoA":0, "HiA":0, "HVal":0, "HEg":0, "LVal":0, "LEg":0, "SVal":0, "ClrH": 0, "ClrL": 0 }, { "Ch":2, "En":1,

```
    "Rng":328,
    "Val":0,
    "Eg":0,
    "Evt":8,
    "LoA":0,
    "HiA":0,
    "HVal":0,
    "HEg":0,
    "LVal":0,
    "LEg":0,
    "SVal":0,
    "ClrH": 0,
    "ClrL": 0
  },
  {
    "Ch":3,
"En":1,
    "Rng":328,
    "Val":0,
    "Eg":0,
    "Evt":0,
    "LoA":0,
    "HiA":0,
    "HVal":0,
    "HEg":0,
    "LVal":0,
    "LEg":0,
    "SVal":0,
    "ClrH": 0,
    "ClrL": 0
  },
  {
    "Ch":4,
"En":1,
    "Rng":328,
    "Val":0,
    "Eg":0,
    "Evt":0,
    "LoA":0,
    "HiA":0,
    "HVal":0,
    "HEg":0,
    "LVal":0,
    "LEg":0,
    "SVal":0,
    "ClrH": 0,
    "ClrL": 0
  }
 ]
}
```


Request : **GET /ai_value/slot_0/ch_2**


```
Content-type: application/json
Response: 200 OK
{
    "Ch":2,
"En":1,
    "Rng":328,
    "Val":0,
    "Eg":0,
    "Evt":8,
    "LoA":0,
    "HiA":0,
    "HVal":0,
```

PUT	None

PATCH	Single/Multi Channel Request:
	PATCH /ai_value/slot_index
	Single Channel Request:
	PATCH /ai_value/slot_index/ch_num
	
	[Example]
	
	Request: **PATCH /ai_value/slot_0**
	
	Content-type: application/json
	{
	"AIVal": [
	{
	"Ch":2,
	"LoA": 0
	},
	{
	"Ch":3,
	"HiA":0
	}
	]
	}
	
	Response: 200 OK
	
	Request: **PATCH /ai_value/slot_0/ch_3**
	
	Content-type: application/json
	{
	"LoA":0
	}
	
	Response: 200 OK

- JSON array name definition:

Field	Abbreviation	Data Type
Array of Analog input configurations	AIVal	Array

- Resource value definitions (Total channels = AI channel number + 1 average channel):

Field	Abbreviation	Data Type	Property	Description
Channel Number	Ch	Number	R	0, 1, …: Analog input channel number. Note for the average channel: The average channel number for a 4-ch AI module is 4.
Input Range	Rng	Number	R	Analog input range. Range code 328 (0x0148) 0 – 10 V 259 (0x0103) +/- 150 mV 260 (0x0104) +/- 500 mV 320 (0x0140) +/- 1 V 321 (0x0141) +/- 2.5 V 322 (0x0142) +/- 5 V 323 (0x0143) +/- 10 V 327 (0x0147) 0 ~ 5 V 384 (0x0180) 4 ~ 20 mA 385 (0x0181) +/- 20 mA 386 (0x0182) 0 ~ 20 mA 65535 Invalid range, if average channel is disable
Channel Enable	En	Number	R	1 / 0: Enable / Disable AI conversion Notice: Average channel is read only. When channel mask of average is not 0, the value is 1.
Channel Raw Value	Val	Number	R	0 ~ 65535:AI measurement data (Raw data)
Channel Engi-neering data	Eg	Number	R	AI engineering data, the value is 1/1000 scale. For example, 1630 → 1.63
Channel Event Status	Evt	Number	R	AI statuses
Low Alarm Status	LoA	Number	RW	Low alarm status Read 1 : low alarm occurred. 0 : not occurred Write 0 : clear the low alarm status
High Alarm Status	HiA	Number	RW	High alarm status Read 1 : high alarm occurred. 0 : not occurred Write 0 : clear the high alarm status
Maximum AI Raw Value	HVal	Number	R	AI max. measurement data (Raw data)

Maximum AI Engineering data	HEg	Number	R	AI max. engineering data, the value is 1/1000 scale For example, 10200→10.2
Minimum AI Raw Value	LVal	Number	R	AI min. measurement data (Raw data)
Minimum AI Engineering data	LEg	Number	R	AI min. engineering data, the value is 1/1000 scale For example, 250 → 0.25
Channel Raw Value After Scaling	SVal	Number	R	0 ~ 65535 : AI measurement data (Raw data) after scaling
Clear Maximum AI Value	ClrH	Number	W	1 : Clear the Maximum AI value
Clear Minimum AI Value	ClrL	Number	W	1 : Clear the Minimum AI value
Physical value after scaling	PEg	Number	R	AI physical value after scaling, the value is 1/1000 scale For example: 150350'150.35
Mapping unit	Uni	String	R	Unit for mapping value Max. 32 characters

* AI Status (2 Registers)			
Lower Register		Higher Register	
Bit	**Description**	**Bit**	**Description**
0	Fail to Provide AI Value	0	DI triggered to Safety Value
1	Over Range	1	DI triggered to Startup Value
2	Under Range	2	Reserved
3	Open Circuit / Burnout	3	Reserved
4	Reserved	4	Reserved
5	Unavailable Channel Configuration	5	Reserved
6	Reserved	6	Reserved
7	ADC Initializing/Error	7	Reserved
8	Reserved	8	Reserved
9	Zero/Span Calibration Error	9	Reserved
10	Reserved	10	Reserved
11	Reserved	11	Reserved
12	Reserved	12	Reserved
13	Reserved	13	Reserved
14	Reserved	14	Reserved
15	Reserved	15	Reserved

D.2.4 RS-485 Port Expansion Data

D.2.4.1 Expansion Bit Data

Table D.4: /expansion_bit/com_x/ch_num	
Description	Retrieves information about the expansion tag bit data resource, the data information is defined by user configuration
URL Structure	**http://10.0.0.1/expansion_bit/com_x (Get only)** **http://10.0.0.1/expansion_bit/com_x /idx_y (Get only)** **http://10.0.0.1/expansion_bit/com_x /ch_num** **where x = 1 ~ : the identifier of COM number** **where y = 0 ~ : the index of data resource requisition, 8 channel in each index** **where num = 0 ~ : the channel number**
HTTP Method	GET: Returns the representation of all of expansion bit data resource. PUT: None PATCH: Apply partial modifications to expansion bit data resource.

GET	Multi-Channel Request? **GET /expansion_bit/com_x** Single Channel Request? **GET /expansion_bit/com_x/ch_num** [Example]: ■ Request : **GET /expansion_bit/com_1/** Content-type: application/json Response: 200 OK { "ExpBit": [{ "Ch":0, "Val":1, "Evt":0, "SID":1, "Addr":1, "MAddr":1001, "WEvt":0 }, { "Ch":1, "Val":0, "Evt":0, "SID":1, "Addr":2, "MAddr":1002, "WEvt":0 }, { "Ch":2, "Val":1, "Evt":0, "SID":1, "Addr":3, "MAddr":1003, "WEvt":0 }, { "Ch":3, "Val":1, "Evt":0, "SID":1, "Addr":4, "MAddr":1004, "WEvt":0 }, { "Ch":4, "Val":1, "Evt":0, "SID":2, "Addr":1, "MAddr":1005, "WEvt":0 }, …… { "Ch":31, "Val":0, "Evt":0, "SID":3, "Addr":17, "MAddr":1032,

JSON array name definition:

Field	Abbreviation	Data Type
Array of Analog input configurations	ExpBit	Array

Resource value definitions:

Field	Abbreviations	Data type	Property	Description
Channel Number	Ch	Number	R	0, 1, …: expansion tag data channel number
Channel Value	Val	Number	R/W	The channel value of expansion tag data Value: 0/1 *After writing action, user must poll the "expansion bit writing status" to get process result
Channel Error Code	Evt	Number	R	The channel error code of expansion tag data (TBD) Modbus exception code Bit 7: Write only mask
Slave ID	SID	Number	R	Modbus RTU slave ID 0~255
Slave Modbus Address	Addr	Number	R	Modbus RTU Slave device polling address: 1-9999
Modbus TCP Mapping Address	MAddr	Number	R	Modbus TCP mapping address of expansion value:1-9999

Event Value (Bit: 6~0)	Description
0 (0x00)	No error
1 (0x01)	Illegal function
2 (0x02)	Illegal data address
3 (0x03)	Illegal data value
4 (0x04)	Slave device failure
5 (0x05)	Acknowledge
6 (0x06)	Slave device busy
7 (0x07)	Negative acknowledge
8(0x08)	Memory parity error
9 (0x09)	Reserved
10 (0x0A)	Gateway path unavailable
11 (0x0B)	Gateway target device failed to respond
12 ~15	Reserved
16 (0x10)	Unavailable
17 (0x11)	Slave response timeout
18 (0x12)	Checksum error
19 (0x13)	Reserved data error
20 (0x14)	Send request fail
21 (0x15)	Unprocessed
22 (0x16)	Read only
23 (0x17)	In processing

<table>
<tr><th>Field</th><th>Abbreviations</th><th>Data type</th><th>Property</th></tr>
<tr><td>Expansion bit writing status</td><td>WEvt</td><td>Number</td><td>The status for preview writing action.

<table><tr><th>Event Value (Bit: 6~0)</th><th>Description</th></tr><tr><td>0 (0x00)</td><td>No error</td></tr><tr><td>1 (0x01)</td><td>Illegal function</td></tr><tr><td>2 (0x02)</td><td>Illegal data address</td></tr><tr><td>3 (0x03)</td><td>Illegal data value</td></tr><tr><td>4 (0x04)</td><td>Slave device failure</td></tr><tr><td>5 (0x05)</td><td>Acknowledge</td></tr><tr><td>6 (0x06)</td><td>Slave device busy</td></tr><tr><td>7 (0x07)</td><td>Negative acknowledge</td></tr><tr><td>8(0x08)</td><td>Memory parity error</td></tr><tr><td>9 (0x09)</td><td>Reserved</td></tr><tr><td>10 (0x0A)</td><td>Gateway path unavailable</td></tr><tr><td>11 (0x0B)</td><td>Gateway target device failed to respond</td></tr><tr><td>12 ~15</td><td>Reserved</td></tr><tr><td>16 (0x10)</td><td>Unavailable</td></tr><tr><td>17 (0x11)</td><td>Slave response timeout</td></tr><tr><td>18 (0x12)</td><td>Checksum error</td></tr><tr><td>19 (0x13)</td><td>Reserved data error</td></tr><tr><td>20 (0x14)</td><td>Send request fail</td></tr><tr><td>21 (0x15)</td><td>Unprocessed</td></tr><tr><td>22 (0x16)</td><td>Read only</td></tr><tr><td>23 (0x17)</td><td>In processing</td></tr></table>
*Event value 0x17: the writing process is not finish, user should poll the status later.</td></tr>
<tr><td colspan="4">Remarks:</td></tr>
<tr><td colspan="4"></td></tr>
</table>

D.2.4.2 Expansion Word Data

/expansion_word/com_x/ch_num

Description	Retrieves information about the expansion tag bit data resource, the data information is defined by user configuration
URL Structure	**http://10.0.0.1/expansion_word/com_x** **http://10.0.0.1/expansion_word/com_x/ch_num** **where x = 1 ~ : the identifier of COM number** **where num = 0 ~ : the channel number**
HTTP Method	GET: Returns the representation of all of expansion bit data resource. PUT: None PATCH: Apply partial modifications to expansion bit data resource.

GET	Multi-Channel Request? **GET /expansion_word/com_x** Single Channel Request? **GET /expansion_word/com_x/ch_num** [Example]: ■ Request : GET /expansion_word/com_1 Content-type: application/json Response: 200 OK { "ExpWord": [{ "Ch":0, "Val":32768, "Evt":0, "SID":1, "Addr":1, "MAddr":1001, "WEvt":0 }, { "Ch":1, "Val":1235, "Evt":0, "SID":1, "Addr":2, "MAddr":1002, "WEvt":0 }, { "Ch":2, "Val":65535, "Evt":0, "SID":1, "Addr":3, "MAddr":1003, "WEvt":0 }, { "Ch":3, "Val":33358, "Evt":0, "SID":1, "Addr":4, "MAddr":1004, "WEvt":0 }, { "Ch":4, "Val":4095, "Evt":0, "SID":2, "Addr":211, "MAddr":1005, "WEvt":0 }, …… { "Ch":31, "Val":0, "Evt":0, "SID":3, "Addr":1, "MAddr":1032,

JSON array name definition:

Field	Abbreviation	Data Type
Array of Analog input configurations	ExpWord	Array

Resource value definitions:

Field	Abbreviations	Data type	Property	Description
Channel Number	Ch	Number	R	0, 1, …: expansion tag data channel number
Channel Value	Val	Number	R/W	The channel value of expansion tag data Value: 0/1 *After writing action, user must poll the "expansion bit writing status" to get process result
Channel Error Code	Evt	Number	R	The channel error code of expansion tag data (TBD) Modbus exception code Bit 7: Write only mask
Slave ID	SID	Number	R	Modbus RTU slave ID 0~255
Slave Modbus Address	Addr	Number	R	Modbus RTU Slave device polling address: 1-9999
Modbus TCP Mapping Address	MAddr	Number	R	Modbus TCP mapping address of expansion value:1-9999

Event Value (Bit: 6~0)	Description
0 (0x00)	No error
1 (0x01)	Illegal function
2 (0x02)	Illegal data address
3 (0x03)	Illegal data value
4 (0x04)	Slave device failure
5 (0x05)	Acknowledge
6 (0x06)	Slave device busy
7 (0x07)	Negative acknowledge
8(0x08)	Memory parity error
9 (0x09)	Reserved
10 (0x0A)	Gateway path unavailable
11 (0x0B)	Gateway target device failed to respond
12 ~15	Reserved
16 (0x10)	Unavailable
17 (0x11)	Slave response timeout
18 (0x12)	Checksum error
19 (0x13)	Reserved data error
20 (0x14)	Send request fail
21 (0x15)	Unprocessed
22 (0x16)	Read only
23 (0x17)	In processing

<table>
<tr><th>Field</th><th>Abbreviations</th><th>Data type</th><th>Property</th></tr>
<tr><td>Expansion bit writing status</td><td>WEvt</td><td>Number</td><td>The status for preview writing action.

<table><tr><th>Event Value (Bit: 6~0)</th><th>Description</th></tr><tr><td>0 (0x00)</td><td>No error</td></tr><tr><td>1 (0x01)</td><td>Illegal function</td></tr><tr><td>2 (0x02)</td><td>Illegal data address</td></tr><tr><td>3 (0x03)</td><td>Illegal data value</td></tr><tr><td>4 (0x04)</td><td>Slave device failure</td></tr><tr><td>5 (0x05)</td><td>Acknowledge</td></tr><tr><td>6 (0x06)</td><td>Slave device busy</td></tr><tr><td>7 (0x07)</td><td>Negative acknowledge</td></tr><tr><td>8(0x08)</td><td>Memory parity error</td></tr><tr><td>9 (0x09)</td><td>Reserved</td></tr><tr><td>10 (0x0A)</td><td>Gateway path unavailable</td></tr><tr><td>11 (0x0B)</td><td>Gateway target device failed to respond</td></tr><tr><td>12 ~15</td><td>Reserved</td></tr><tr><td>16 (0x10)</td><td>Unavailable</td></tr><tr><td>17 (0x11)</td><td>Slave response timeout</td></tr><tr><td>18 (0x12)</td><td>Checksum error</td></tr><tr><td>19 (0x13)</td><td>Reserved data error</td></tr><tr><td>20 (0x14)</td><td>Send request fail</td></tr><tr><td>21 (0x15)</td><td>Unprocessed</td></tr><tr><td>22 (0x16)</td><td>Read only</td></tr><tr><td>23 (0x17)</td><td>In processing</td></tr></table>
*Event value 0x17: the writing process is not finish, user should poll the status later.</td></tr>
<tr><td colspan="4">Remarks:</td></tr>
<tr><td colspan="4"></td></tr>
</table>

D.2.5 Data Logger

D.2.5.1 /log_message

Description	Retrieves the log data in system memory.
URL Structure	http://10.0.0.1/log_message
HTTP Method	GET: According to the setting of filtering, server returns the all/partial of logged data.

GET	Request:
	GET /log_message
	
	[Example]:
	Request: **GET /log_message** for WISE-4060/LAN module
	
	Content-type: application/json
	Response: 200 OK
	{
	"LogMsg": [
	{
	"PE":128,
	"TIM":"2014-11-11T15:48:32+08:00",
	"UID":"ADAM-4060/LAN_00D0C9FE1601",
	"MAC":"00-D0-C9-FE-16-01",
	"Record" :
	[
	[0,3,3,1],
	[0,2,4,150],
	[0,5,5,250]
	]
	},
	{
	"PE":128,
	"TIM":"2014-11-11T15:49:44+08:00",
	"UID":"ADAM-4060/LAN_00D0C9FE1601",
	"MAC":"00-D0-C9-FE-16-01",
	"Record" :
	[
	[0,3,3,0],
	[0,2,4,140],
	[0,5,5,240]
	]
	},
	{
	"PE":128,
	"TIM":"2014-11-11T15:51:02+08:00",
	"UID":"ADAM-4060/LAN_00D0C9FE1601",
	"MAC":"00-D0-C9-FE-16-01",
	"Record" :
	[
	[0,3,3,0],
	[0,2,4,130],
	[0,5,5,230]
	]
	}
	]
	}

JSON array name definition:

Field	Abbreviation	Data Type
Array of log messages	LogMsg	Array
Array of I/O records	Record	Array

Resource value definitions:

Field	Abbreviations	Data type	Property	Description
Periodic/Event	128	Number	R	Recording mode of storage 1 DI 2 DO 4 Event from AI 8 AO 16 WDT 128 Periodic
Timestamp	TIM	String	R	Timestamp of the storage “Coordinated Universal Time (UTC) Ex. “1415757750” corresponds to November 12, 2014, 2:02:30 am, Standard Time. (meanwhile, 2014, 10:02:30 am, Taipei Time.) “Local Date/Time according GMT time zone (ISO 8601) Ex. “1994-11-05T08:15:30-05:00” corresponds to November 5, 1994, 8:15:30 am, US Eastern Standard Time.
UUID	UID	String	R	Universally Unique Identifier (UUID) Max. 32 characters
MAC ID	MAC	String	R	MAC address. (12+5) characters, ex, “00-D0-C9-F0-63-F7

<table>
<tr>
<td>Recording message</td>
<td>Record</td>
<td>Array</td>
<td>R</td>
<td>* The information in array is as follows.
[Slot-index, Channel-index, I/O-type-index, I/O-value]
* The data type in array is as follows.
[Number, Number, Number, Number]
Notice: When the I/O-type-index is engineering type (12, 13, 14, 18), the I/O value is 1/1000 scale.

<table>
<tr><th>Index</th><th>Recording I/O-type of the storage</th></tr>
<tr><td>0</td><td>Invalid</td></tr>
<tr><td>1</td><td>DI Logic Status</td></tr>
<tr><td>2</td><td>DI Counter value</td></tr>
<tr><td>3</td><td>DI Frequency value</td></tr>
<tr><td>4</td><td>DO Logic Status</td></tr>
<tr><td>5</td><td>DO Absolute Pulse Output value</td></tr>
<tr><td>6</td><td>DO Incremental Pulse Output Value</td></tr>
<tr><td>7</td><td>AI value</td></tr>
<tr><td>8</td><td>Historical Maximum AI value</td></tr>
<tr><td>9</td><td>Historical Minimum AI value</td></tr>
<tr><td>10</td><td>AI value after scaling</td></tr>
<tr><td>11</td><td>AI status flags</td></tr>
<tr><td>12</td><td>AI engineering value</td></tr>
<tr><td>13</td><td>Historical Maximum AI engineering value</td></tr>
<tr><td>14</td><td>Historical Minimum AI engineering value</td></tr>
<tr><td>15</td><td>AO value</td></tr>
<tr><td>16</td><td>AO value after scaling</td></tr>
<tr><td>17</td><td>AO status flags</td></tr>
<tr><td>18</td><td>AO engineering value</td></tr>
<tr><td>19</td><td>AI physical value</td></tr>
<tr><td>20</td><td>AI engineering value (floating type)</td></tr>
<tr><td>21</td><td>Historical Maximum AI engineering value (floating type)</td></tr>
<tr><td>22</td><td>Historical Minimum AI engineering value (floating type)</td></tr>
<tr><td>23</td><td>AI physical value (floating tpye)</td></tr>
<tr><td>30</td><td>Expansion bit data</td></tr>
<tr><td>31</td><td>Expansion bit error code</td></tr>
<tr><td>32</td><td>Expansion word data</td></tr>
</table>
</td>
</tr>
<tr>
<td>Remarks</td>
<td colspan="4"></td>
</tr>
</table>

Table D.5: WISE-4250 supported Ciphers List

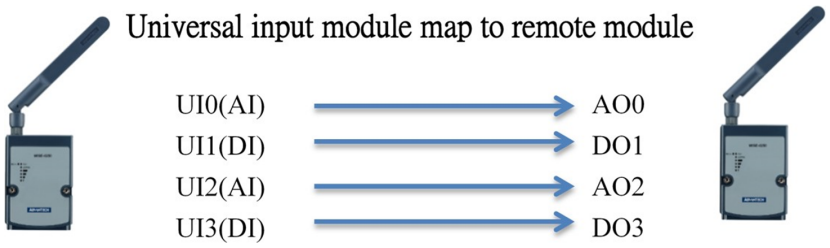
Module	Security	Ciphers
WISE-4250	TLS 1.2 ~ 1.3	TLS_AES_128_GCM_SHA256 TLS_AES_128_CCM_SHA256 TLS_AES_128_CCM_8_SHA256 TLS_ECDHE_ECDSA_WITH_AES_256_CCM TLS_DHE_RSA_WITH_AES_256_CCM TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA TLS_DHE_RSA_WITH_AES_256_CBC_SHA TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8 TLS_DHE_RSA_WITH_AES_256_CCM_8 TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA256 TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_ECDSA_WITH_AES_128_CCM TLS_DHE_RSA_WITH_AES_128_CCM TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA TLS_DHE_RSA_WITH_AES_128_CBC_SHA TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 TLS_DHE_RSA_WITH_AES_128_CCM_8 TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_GCM_SHA256 TLS_ECDHE_RSA_WITH_CAMELLIA_128_GCM_SHA256 TLS_DHE_RSA_WITH_CAMELLIA_128_GCM_SHA256 TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_CBC_SHA256 TLS_ECDHE_RSA_WITH_CAMELLIA_128_CBC_SHA256 TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA256 TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA TLS_ECDHE_ECDSA_WITH_ARIA_128_GCM_SHA256 TLS_ECDHE_RSA_WITH_ARIA_128_GCM_SHA256 TLS_DHE_RSA_WITH_ARIA_128_GCM_SHA256 TLS_ECDHE_ECDSA_WITH_ARIA_128_CBC_SHA256 TLS_ECDHE_RSA_WITH_ARIA_128_CBC_SHA256 TLS_DHE_RSA_WITH_ARIA_128_CBC_SHA256 TLS_RSA_WITH_AES_256_CCM TLS_RSA_WITH_AES_256_CBC_SHA256 TLS_RSA_WITH_AES_256_CBC_SHA TLS_ECDH_RSA_WITH_AES_256_CBC_SHA TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA TLS_RSA_WITH_AES_256_CCM_8 TLS_RSA_WITH_CAMELLIA_256_CBC_SHA256 TLS_RSA_WITH_CAMELLIA_256_CBC_SHA TLS_RSA_WITH_AES_128_GCM_SHA256 TLS_RSA_WITH_AES_128_CCM TLS_RSA_WITH_AES_128_CBC_SHA256 TLS_RSA_WITH_AES_128_CBC_SHA TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256 TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256 TLS_ECDH_RSA_WITH_AES_128_CBC_SHA TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256 TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256 TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA TLS_RSA_WITH_AES_128_CCM_8 TLS_RSA_WITH_CAMELLIA_128_GCM_SHA256 TLS_RSA_WITH_CAMELLIA_128_CBC_SHA256 TLS_RSA_WITH_CAMELLIA_128_CBC_SHA TLS_ECDH_RSA_WITH_CAMELLIA_128_GCM_SHA256 TLS_ECDH_RSA_WITH_CAMELLIA_128_CBC_SHA256 TLS_ECDH_ECDSA_WITH_CAMELLIA_128_GCM_SHA256 TLS_ECDH_ECDSA_WITH_CAMELLIA_128_CBC_SHA256 TLS_ECDH_ECDSA_WITH_ARIA_128_GCM_SHA256 TLS_ECDH_RSA_WITH_ARIA_128_GCM_SHA256 TLS_RSA_WITH_ARIA_128_GCM_SHA256 TLS_ECDH_ECDSA_WITH_ARIA_128_CBC_SHA256 TLS_ECDH_RSA_WITH_ARIA_128_CBC_SHA256 TLS_RSA_WITH_ARIA_128_CBC_SHA256

D.2.6 Peer to Peer (P2P)

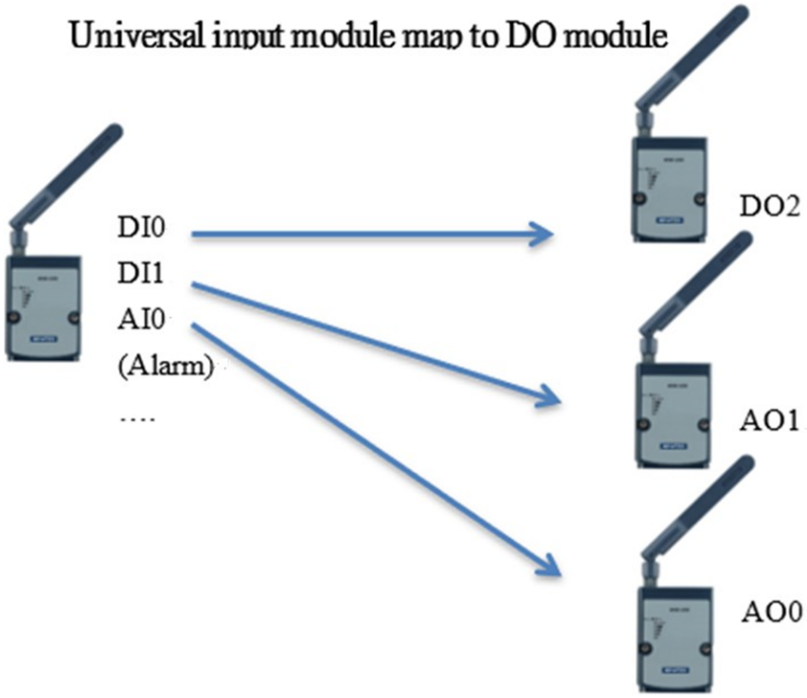
Peer-to-peer (P2P), is a computing or networking distributed application architecture that partitions tasks or workloads among peers.

It defines a mapping between input module and output module, updating remote module output periodically and change of status.

WISE-4000/4200 Wi-Fi series all support basic mode for one target module,



or advanced mode for multiple target modules



with 5 kind of channel mapping: DI map to DO

ADVANTECH

Enabling an Intelligent Planet

www.advantech.com

Please verify specifications before quoting. This guide is intended for reference purposes only.

All product specifications are subject to change without notice.

No part of this publication may be reproduced in any form or by any means, electronic, photocopying, recording or otherwise, without prior written permission of the publisher.

All brand and product names are trademarks or registered trademarks of their respective companies.

© Advantech Co., Ltd. 2024