

## Introduction

The Simple Network Management Protocol (SNMP) is an important communication protocol for network device management. It can be used to detect and isolate malicious wireless LAN access points, so that to protect internal important data from leaking or external deliberate attack to guarantee information security.

This document introduces the architecture and usage method of SNMP to help users to develop a network management system as needed.

*Note: The corresponding code in this application note is developed on the basis of V2.x.x BSP provided by Artery. For other versions of BSP, please pay attention to the differences in usage.*

Applicable products:

Part number	AT32F407xx
-------------	------------

# Contents

<b>1</b>	<b>Overview .....</b>	<b>4</b>
1.1	SNMP components.....	4
1.2	SNMP architecture .....	4
1.2.1	Master agent .....	4
1.2.2	Subagent .....	4
1.2.3	Management station.....	5
1.3	SNMP principle and evolution.....	5
<b>2</b>	<b>Agent port software configuration.....</b>	<b>7</b>
2.1	RL-TCPnet .....	7
2.2	SNMP Agent.....	9
2.3	SNMP management station.....	10
2.3.1	Software requirement.....	10
<b>3</b>	<b>Revision history.....</b>	<b>12</b>

## List of figures

Figure 1. MIB hierarchical scheme .....	5
Figure 2. SNMP communication architecture .....	6
Figure 3. Macro definitions of Ethernet-related parameters.....	7
Figure4. Call RL-TCPnet initialization and poll API in main function and main loop.....	8
Figure 5. Ping evaluation board through PC .....	8
Figure 6. SNMP MIB .....	10
Figure 7. Access MIB through SNMP command .....	11

# 1 Overview

## 1.1 SNMP components

The SNMP consists of:

1. Network Management Systems (NMSs)
2. Managed Device
3. Agent

The network management system is an executable program, which is used to monitor and control the device under management. It can be called managing entity where the network administrator interacts with network devices. It also provides a large number of computing and storage resources for network management. A managed network may have more than one network management system.

A managed device is a network node that contains an SNMP agent that exists in the managed network. It collects and stores management information through the Management Information Base (MIB), and allows the network management system to access this information through SNMP agent.

The agent is a network management software module that is packaged within the managed device. It controls the management information of local machine and transmits this information in a SNMP-compatible format.

## 1.2 SNMP architecture

In terms of architecture, SNMP framework consists of master agent, subagent and management station.

### 1.2.1 Master agent

The master agent is a software that runs on an executable SNMP network component, which can respond to the SNMP request received from management station. It functions like the Server in the Client/Server architecture. The master agent relies on the subagents to provide administrative information about particular functions.

When the system has multiple manageable subsystems, the master agent transmits the request(s) received from one or more subagents. These subagent model the specified objects in a subsystem and in the interface that monitors and manages this subsystem. The master agent and subagent can be combined into the so-called "agent".

### 1.2.2 Subagent

A subagent is software that runs on an executable SNMP network component, which performs the defined information and management functions in the MIB of specified subsystem. Its main features include:

1. Collect the master agent data
2. Configure the master agent parameters
3. Respond to the request from administrator
4. Generate a warning or trap

Proper separation of protocol and management data structure makes it easy to use SNMP to monitor and manage various different subsystems within the same network. The MIB model runs all

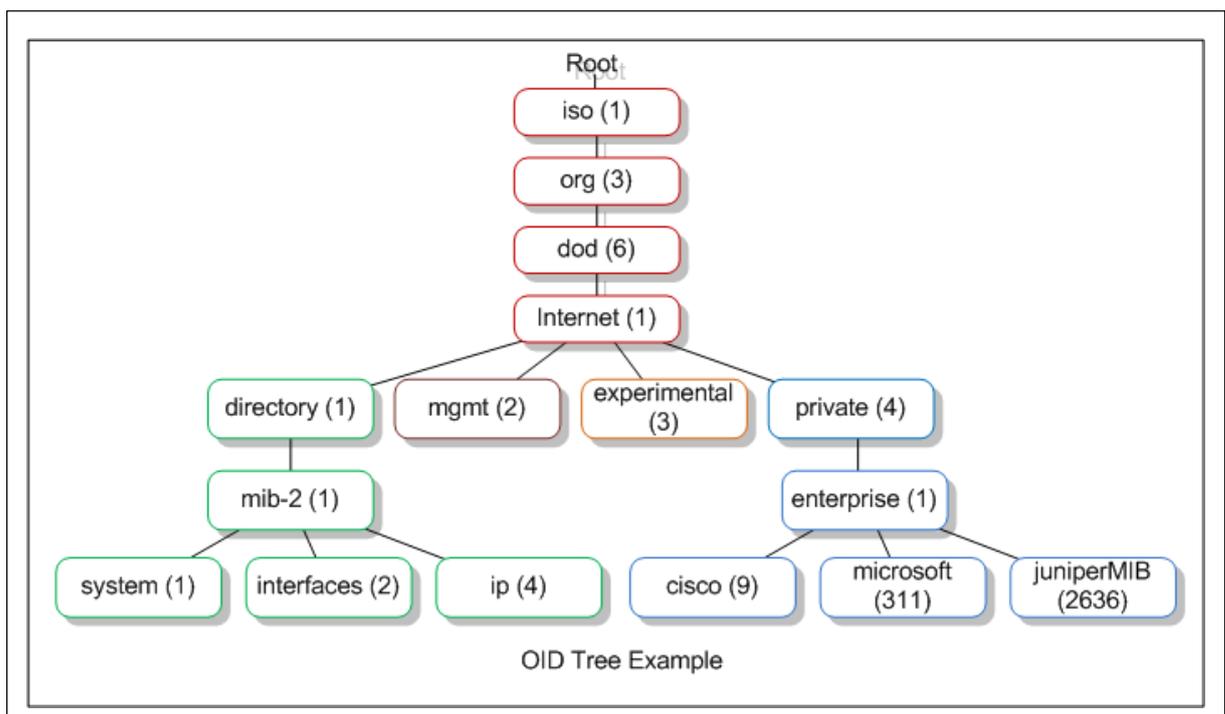
layers of the OSI reference model and can be extended to applications such as databases, E-mail, and the J2EE reference model.

### 1.2.3 Management station

The administrator or management station serves the third element of SNMP framework. It functions like the Client in the Client/Server architecture. The management station sends a request for administrative operation according to the behavior of the administrator or application, and receives the TRAP from the agent.

SNMP Agent presents the relevant information of the managed device in the form of variables. Each variable has a unique object identifier (OID) that is described in hierarchy mode in MIB. For example, OID 1.3.6.1.4.1.9 represents Cisco company.

Figure 1. MIB hierarchical scheme

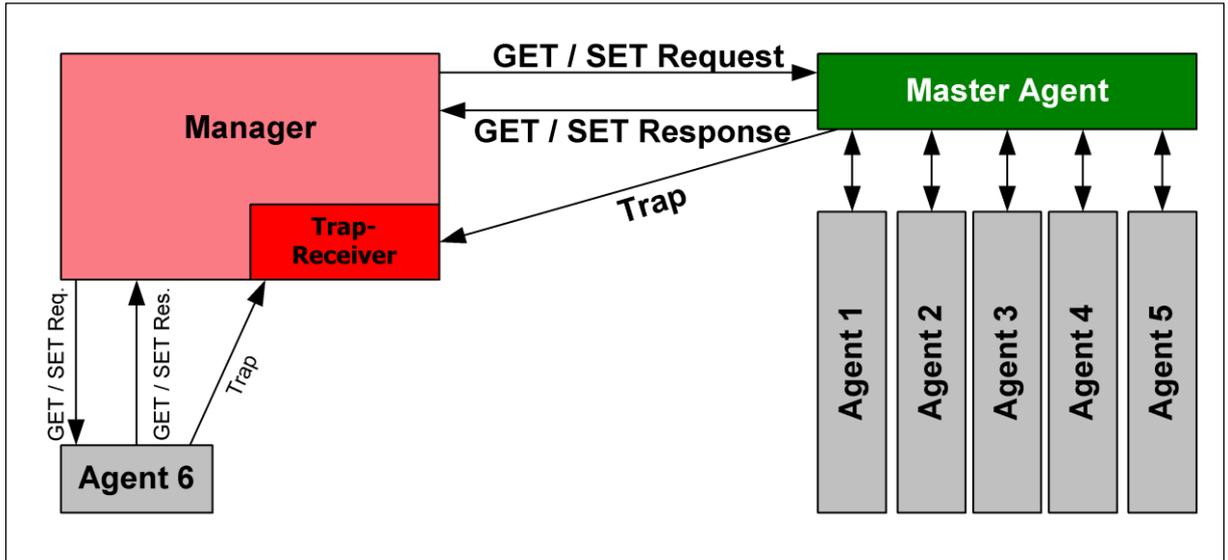


## 1.3 SNMP principle and evolution

SNMP is an application layer operating on the OSI model. The management port sends the request to the agent through UDP (port 161), and the agent sends the response to the management port through the source port. In addition, in case of any abnormal event of the monitored device, such as cold start or link down, the agent actively sends a notification to the management port (port 162) through UDP.

The management port can retrieve the information about the monitored device from the agent through the Get, GetNext or GetBulk commands, and also actively transmit data through the Trap or Inform commands. In addition, the management port achieves the purpose of system management through Set command.

Figure 2. SNMP communication architecture



## 2 Agent port software configuration

This sample code uses AT32F407 as the Agent, MAC as Ethernet interface, and RL-TCPnet (provided by KEIL) as the protocol stack, and the SNMP is located in the application layer of OSI model. This section introduces how to use the RL-TCPNET-based SNMP Agent.

### 2.1 RL-TCPnet

RL-TCPnet is a component of RL-ARM library, and an implementation of TCP/IP protocol stack. The stack is designed to reduce memory usage and code size, which makes it suitable for devices with limited resources, such as embedded systems. The RL-TCPnet library is basic implementation of the software routine of ARM7, ARM9, Cortex-M1 and Cortex-M3.

This document uses RL-TCPnet as the TCP/IP protocol stack, which makes easier to realize Ethernet communication than LwIP, as long as the library provided by Keil is packaged into the project file. This section introduces how to configure parameters to guarantee normal Ethernet communication after the RL-TCPnet is packaged into the project.

Ethernet-related macro definitions are placed in Net\_Config.c. Configure as follows:

1. Set the Ethernet enable macro definition to 1: `#define ETH_ENABLE 1`
2. Set the MAC Address macro, and the macro definition in code is “\_MACx, x = 1-6”
3. Set the IP Address macro, and the macro definition in code is “\_IPx, x = 1-4”
4. Set the macro for the subnet mask, and the macro definition in code is “\_MSKx, x = 1-4”
5. Set the macro for gateway address, and the macro definition in code is “\_GWx, x = 1-4”
6. Call “init\_TcpNet()” in the main function to initialize RL-TCPnet protocol
7. Call “main\_TcpNet()” in the main function to poll Ethernet-related features (eth\_run\_link(), ip\_run\_local(), icmp\_run\_engune(), etc.)

Figure 3. Macro definitions of Ethernet-related parameters

```

41 // <i> Enable or disable Ethernet Network Interface
42 #define ETH_ENABLE 1
43
44 // <h>MAC Address
45 // =====
46 // <i> Local Ethernet MAC Address
47 // <i> Value FF:FF:FF:FF:FF:FF is not allowed.
48 // <i> It is an ethernet Broadcast MAC address.
49 // <o>Address byte 1 <0x00-0xff:2>
50 // <i> LSB is an ethernet Multicast bit.
51 // <i> Must be 0 for local MAC address.
52 // <i> Default: 0x00
53 #define _MAC1 0x1E
54
55 // <o>Address byte 2 <0x00-0xff>
56 // <i> Default: 0x30
57 #define _MAC2 0x30
58
59 // <o>Address byte 3 <0x00-0xff>
60 // <i> Default: 0x6C
61 #define _MAC3 0x6C
62
63 // <o>Address byte 4 <0x00-0xff>
64 // <i> Default: 0x00
65 #define _MAC4 0xA2
66
67 // <o>Address byte 5 <0x00-0xff>
68 // <i> Default: 0x00
69 #define _MAC5 0x45
70
71 // <o>Address byte 6 <0x00-0xff>
72 // <i> Default: 0x01
73 #define _MAC6 0x5E

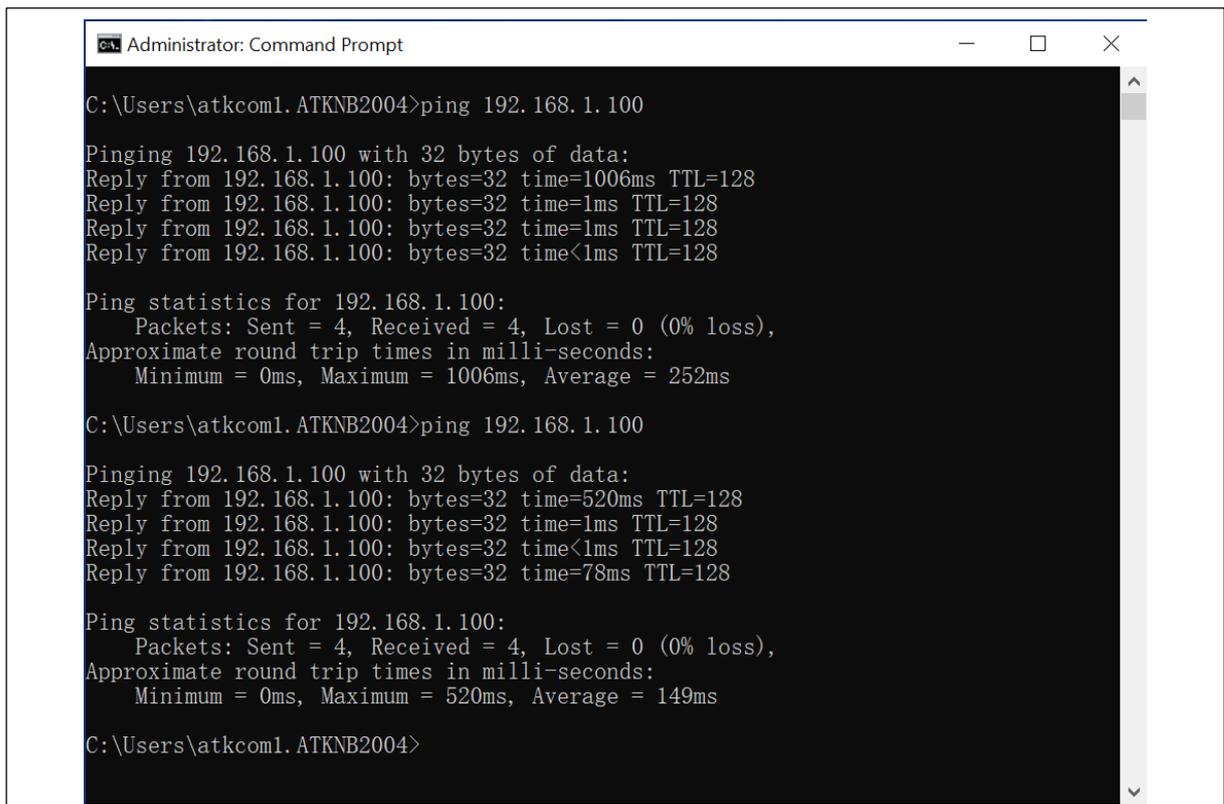
```

Figure4. Call RL-TCPnet initialization and poll API in main function and main loop

```
52  /* Gloable functions -----*/
53  /**
54   * @brief Main Function.
55   * @param None
56   * @retval None
57   */
58  int main(void)
59  {
60   /* -----BSP Init -----*/
61   AT32_Board_Init();
62   UART_Print_Init(115200);
63   /* Setup AT32 system (clocks, Ethernet, GPIO, NVIC) */
64   System_Setup();
65   Delay_init();
66
67   init_TcpNet();
68   while(1)
69   {
70     timer_tick();
71     main_TcpNet();
72   }
73 }
```

After the preceding steps are complete, the TCP/IP protocol stack is connected. At this point, connect to PC and verify through ping.

Figure 5. Ping evaluation board through PC



```
Administrator: Command Prompt
C:\Users\atkcom1.ATKNB2004>ping 192.168.1.100

Pinging 192.168.1.100 with 32 bytes of data:
Reply from 192.168.1.100: bytes=32 time=1006ms TTL=128
Reply from 192.168.1.100: bytes=32 time=1ms TTL=128
Reply from 192.168.1.100: bytes=32 time=1ms TTL=128
Reply from 192.168.1.100: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.100:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1006ms, Average = 252ms

C:\Users\atkcom1.ATKNB2004>ping 192.168.1.100

Pinging 192.168.1.100 with 32 bytes of data:
Reply from 192.168.1.100: bytes=32 time=520ms TTL=128
Reply from 192.168.1.100: bytes=32 time=1ms TTL=128
Reply from 192.168.1.100: bytes=32 time<1ms TTL=128
Reply from 192.168.1.100: bytes=32 time=78ms TTL=128

Ping statistics for 192.168.1.100:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 520ms, Average = 149ms

C:\Users\atkcom1.ATKNB2004>
```

## 2.2 SNMP Agent

To use the SNMP Agent feature, open the `SNMP_ENABLE` macro in `Net_Config.c` to start initialization for SNMP-related modules.

The most important component of SNMP module is MIB (Management Information Base). MIB-related codes are placed in `snmp_mib.c`, and MIB provides the data to be accessed by the management station. In the code, MIB is constructed as a heterogeneous array, with each element as a structure that includes Object Type, Object ID length, Object ID value, Size of a variable, Pointer to a variable, and Write/Read event callback function. Fill in the required information according to the elements of the structure, and the data can be accessed by the management station.

The MIB in code can be divided into two parts. The first part is System MIB, which belongs to the block of the protocol, and some attributes can be changed, such as the string or value of the response, but most attributes cannot be modified. The other part is Experimental MIB, where user defined MIB elements can be added, such as the on-board LED switch through the management station.

Figure 6. SNMP MIB

```

40 const MIB_ENTRY snmp_mib[] = {
41
42     /* ----- System MIB ----- */
43
44     /* SysDescr Entry */
45     { MIB_OCTET_STR | MIB_ATR_RO,
46       8, {OID0(1,3), 6, 1, 2, 1, 1, 1, 0},
47       MIB_STR("Embedded System SNMP V1.0"),
48       NULL },
49
50     /* SysObjectID Entry */
51     { MIB_OBJECT_ID | MIB_ATR_RO,
52       8, {OID0(1,3), 6, 1, 2, 1, 1, 2, 0},
53       MIB_STR("\x2b\x06\x01\x02\x01\x01\x02\x00"),
54       NULL },
55
56     /* SysUpTime Entry */
57     { MIB_TIME_TICKS | MIB_ATR_RO,
58       8, {OID0(1,3), 6, 1, 2, 1, 1, 3, 0},
59       4, &snmp_SysUpTime,
60       NULL },
61
62     /* SysContact Entry */
63     { MIB_OCTET_STR | MIB_ATR_RO,
64       8, {OID0(1,3), 6, 1, 2, 1, 1, 4, 0},
65       MIB_STR("test@arterytek.com"),
66       NULL },
67
68     /* SysName Entry */
69     { MIB_OCTET_STR | MIB_ATR_RO,
70       8, {OID0(1,3), 6, 1, 2, 1, 1, 5, 0},
71       MIB_STR("Evaluation board"),
72       NULL },
73
74     /* SysLocation Entry */
75     { MIB_OCTET_STR | MIB_ATR_RO,
76       8, {OID0(1,3), 6, 1, 2, 1, 1, 6, 0},
77       MIB_STR("Local"),
78       NULL },
79
80     /* SysServices Entry */
81     { MIB_INTEGER | MIB_ATR_RO,
82       8, {OID0(1,3), 6, 1, 2, 1, 1, 7, 0},
83       MIB_INT(sysServices),
84       NULL },
85
86     /* End of System MIB */
87     { MIB_OCTET_STR | MIB_ATR_RO,
88       8, {OID0(1,3), 6, 1, 2, 1, 1, 8, 0},
89       MIB_STR(""),
90       NULL },
91
92     /* End of MIB */
93     { MIB_OCTET_STR | MIB_ATR_RO,
94       8, {OID0(1,3), 6, 1, 2, 1, 1, 9, 0},
95       MIB_STR(""),
96       NULL }
97 }

```

## 2.3 SNMP management station

### 2.3.1 Software requirement

#### 2.3.1.1 Python

After the SNMP Agent is created, a SNMP management station is required for communication. Users can download a proper software from website for communication with SNMP Agent. This document uses SNMP tool that can be executed in Python. Please download Python from <https://www.python.org>.

#### 2.3.1.2 SNMP Tool

It is a collection of command-line SNMP tools written in Python and integrated with the standard

SNMP tools that come with Net-SNMP, such as snmpget, snmpwalk, etc.

In most cases, users can switch between the Net-SNMP tool and the tools provided by this project, without making too many changes on the command line.

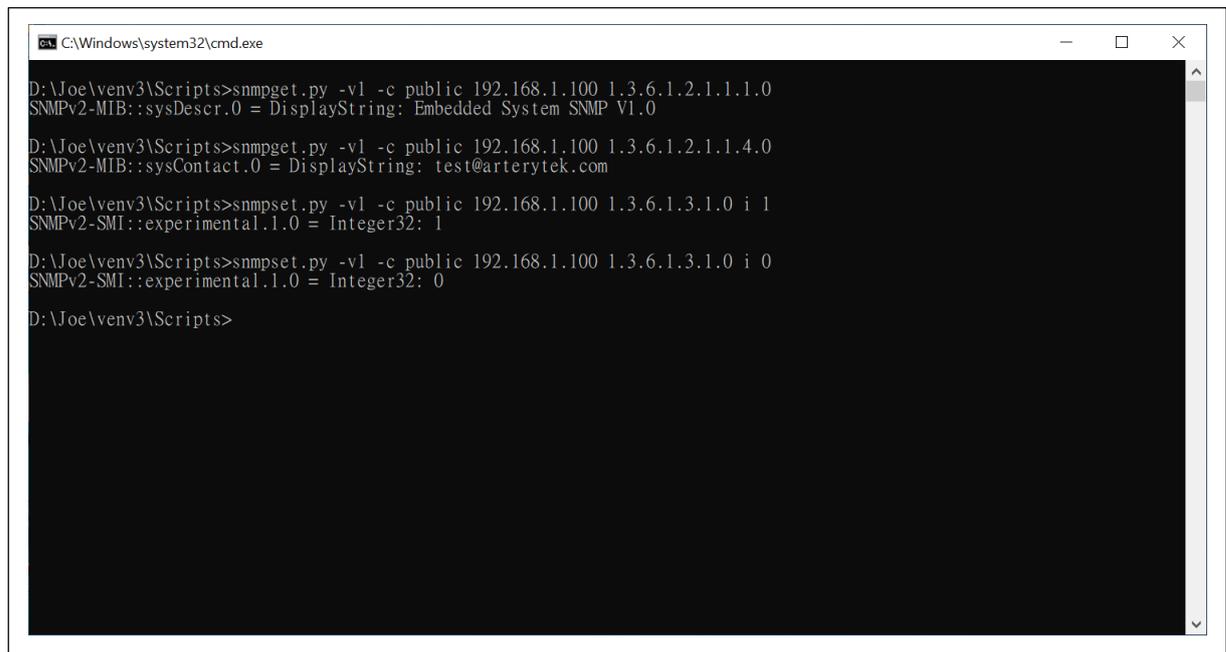
The goal of the project is to bring SNMP tools to a wider range of computing platforms and to introduce new SNMP features using a high-level programming language.

SNMP tools are free and open source. The source code is hosted in the GitHub repository and distributed under a 2-clause BSD License.

Users can download this tool from official website, and install as instructed. If necessary, go to Python community to download the required dependence kits.

After successful setup, users can use SNMP command for communication with SNMP Agent.

**Figure 7. Access MIB through SNMP command**



```
C:\Windows\system32\cmd.exe
D:\Joe\venv3\Scripts>snmpget.py -v1 -c public 192.168.1.100 1.3.6.1.2.1.1.0
SNMPv2-MIB::sysDescr.0 = DisplayString: Embedded System SNMP V1.0
D:\Joe\venv3\Scripts>snmpget.py -v1 -c public 192.168.1.100 1.3.6.1.2.1.1.4.0
SNMPv2-MIB::sysContact.0 = DisplayString: test@arterytek.com
D:\Joe\venv3\Scripts>snmpset.py -v1 -c public 192.168.1.100 1.3.6.1.3.1.0 i 1
SNMPv2-SMI::experimental.1.0 = Integer32: 1
D:\Joe\venv3\Scripts>snmpset.py -v1 -c public 192.168.1.100 1.3.6.1.3.1.0 i 0
SNMPv2-SMI::experimental.1.0 = Integer32: 0
D:\Joe\venv3\Scripts>
```

### 3 Revision history

Table 1. Document revision history

Date	Version	Revision note
2021.12.07	2.0.0	Updated codes and content to V2.
2022.03.04	2.0.1	Modified the table of contents.

**IMPORTANT NOTICE – PLEASE READ CAREFULLY**

Purchasers are solely responsible for the selection and use of ARTERY's products and services, and ARTERY assumes no liability whatsoever relating to the choice, selection or use of the ARTERY products and services described herein.

No license, express or implied, to any intellectual property rights is granted under this document. If any part of this document deals with any third party products or services, it shall not be deemed a license grant by ARTERY for the use of such third party products or services, or any intellectual property contained therein, or considered as a warranty regarding the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

Unless otherwise specified in ARTERY's terms and conditions of sale, ARTERY provides no warranties, express or implied, regarding the use and/or sale of ARTERY products, including but not limited to any implied warranties of merchantability, fitness for a particular purpose (and their equivalents under the laws of any jurisdiction), or infringement of any patent, copyright or other intellectual property right.

Purchasers hereby agrees that ARTERY's products are not designed or authorized for use in: (A) any application with special requirements of safety such as life support and active implantable device, or system with functional safety requirements; (B) any air craft application; (C) any automotive application or environment; (D) any space application or environment, and/or (E) any weapon application. Purchasers' unauthorized use of them in the aforementioned applications, even if with a written notice, is solely at purchasers' risk, and is solely responsible for meeting all legal and regulatory requirement in such use.

Resale of ARTERY products with provisions different from the statements and/or technical features stated in this document shall immediately void any warranty grant by ARTERY for ARTERY products or services described herein and shall not create or expand in any manner whatsoever, any liability of ARTERY.