Application Note:

Configuring the SNMP Extended Agent for xPico 200 Series and XPort EDGE

Part Number APP-0049
Revision A August 2020

Intellectual Property

Contacts

Lantronix, Inc.
7535 Irvine Center Drive, Suite 100
Irvine, CA 92618, USA

Toll Free:	800-526-8766
Phone:	949-453-3990
Fax:	949-453-3995

Technical Support
Online: www.lantronix.com/support

Sales Offices
For a current list of our domestic and international sales offices, go to the Lantronix web site at www.lantronix.com/about/contact

Disclaimer

Revision History

Date	Rev.	Comments
July 2020	A	Initial document.

For the latest revision of this product document, please check our online documentation at www.lantronix.com/support/documentation.

Contents

Introduction

The xPico 200 series SDK provides a means for an application developer to add standard MIBs by loading customized code into the xPico 200. Alternatively, to add other MIBs, the application developer may enable the SNMPv2 extended agent and write their own customized code in an attached host processor as per the protocol.

This document describes how to configure the xPico 200 or XPort EDGE to work with an attached host processor to extend the SNMPv2 agent capabilities. The examples and description refer to xPico 200, but apply to XPort EDGE devices as well.

Related Documentation

- xPico 200 Series SDK User Guide (HTML)
- XPort EDGE SDK User Guide (HTML)

Required Firmware

SNMP is not included in the default firmware. To use SNMP, use the SDK to build a firmware image that includes the SNMP module. The firmware version should be:

- xPico 200 series - 4.2.0.0R7 or greater
- XPort EDGE - 4.2.0.0R7 or greater

The SNMP extended agent applies to the following products:

- xPico 240
- xPico 250
- xPico 270
- XPort EDGE

SNMP Extended Agent Architecture

SNMP architecture consists of three layers: the SNMP network manager, the SNMP agent, and MIBs. The following figure shows the Lantronix SNMP extended agent architecture in the context of these three layers.

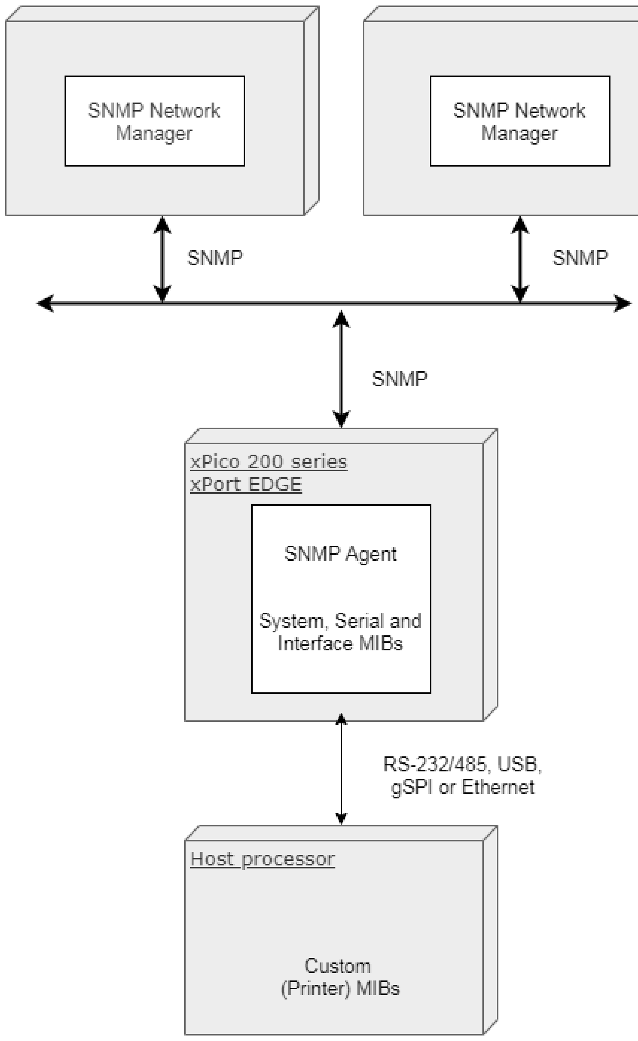


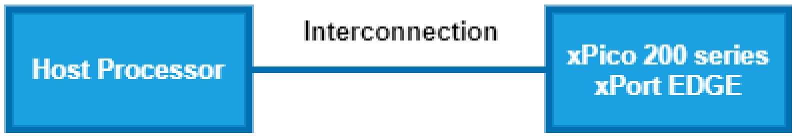
Figure 1. SNMP Agent Extended Architecture

The SNMP network manager will query the Lantronix SNMP agent through SNMP protocol. The SNMP agent is built into the xPico 200 firmware as an optional SDK feature by the OEM application developer. The OEM application developer writes and loads customized code for the host processor that includes the custom MIBs. The following sections describe setup details and behavior of the SNMP agent and SNMP extended agent.

General Setup

The host processor connects to the xPico 200 by any physical interface that supports at least one xPico 200 Line. This can be:

- RS-232/RS-485
- USB
- gSPI
- Ethernet



The Line protocol may be set to

a. **Tunnel**, for simplicity if a Line can be dedicated for this purpose, or
b. **Mux**, which allows for multi-purpose sharing of the Line.

The host processor accepts a TCP connection from the xPico 200. The listen port for this TCP connection must be configured in the xPico 200 as the *Host Port*. The xPico 200 will send all the received SNMP packets to the host over this TCP connection. The host processor will examine each forwarded packet and decide to either

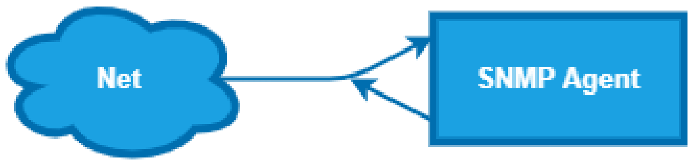
a. let the Lantronix SNMP agent handle it, or
b. compose its own reply.

The host sends its responses back over the same TCP connection. There is one host response per incoming SNMP packet.

SNMP Agent Behavior

By default, the Lantronix SNMP agent responds directly to all SNMP packets. This will be the behavior until the extended agent is configured.

The following figure shows the behavior of the original Lantronix SNMP agent.



SNMP Extended Agent Behavior

The extended agent reads the inbound packets and decides whether to handle the data in the host or to send it to the Lantronix SNMP agent. To configure the extended agent on the xPico 200:

1. configure a "Host Port" (not <None>), and
2. accept a TCP connection on that port for TCP messages.

The following figure shows the behavior of the extended agent.

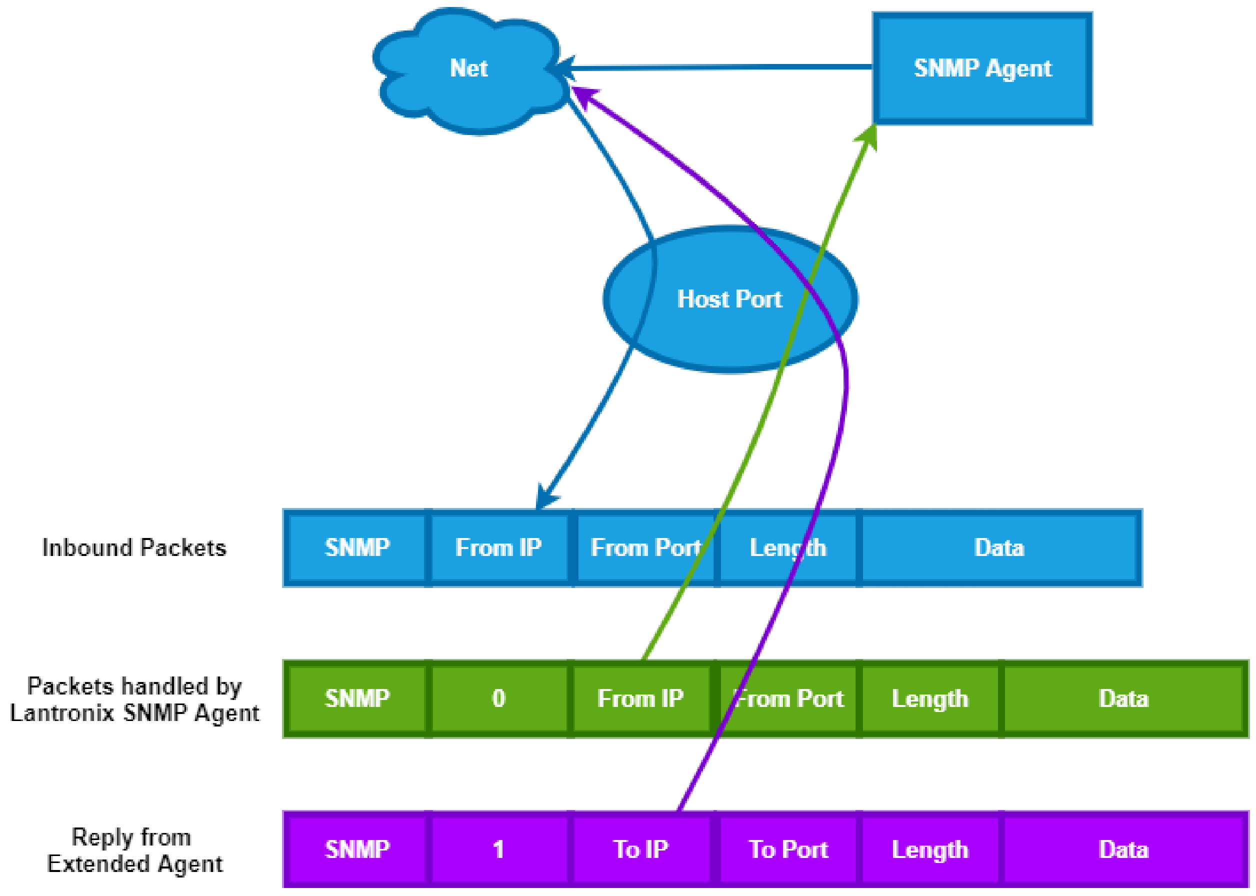


Figure 2. SNMP Extended Agent proprietary protocol

Inbound Packets

All incoming SNMP packets are forwarded via the loopback interface (lo0) to the TCP Host Port on the xPico 200. Each has a header consisting of:

1. "SNMP" (4 ASCII bytes)
2. From IP (4 binary bytes, high order first)
3. From Port (2 binary bytes, high order first)
4. Length (2 binary bytes, high order first)

The header is followed by the Data field which consists of Length bytes. This is the payload from the original inbound packet, with the Ethernet and IP headers removed.

The host processor examines each of these inbound packets and, for each inbound packet, must decide either to allow the Lantronix SNMP agent to reply or to reply on its own to extend the SNMP agent.

Packets to be Handled by Lantronix SNMP Agent

For each packet to be handled by Lantronix, the host processor essentially echoes back the packet, but the header has an additional one-byte binary "0" after the ASCII "SNMP".

Reply from Extended Agent

Otherwise, the host processor may compose its own response to the SNMP packet. The host replies with a header consisting of:

1. "SNMP" (4 ASCII bytes)
2. "1" (1 binary byte)
3. To IP (4 binary bytes, high order first)
4. To Port (2 binary bytes, high order first)
5. Length (2 binary bytes, high order first)

This header is followed by the payload for the UDP response. This payload field does not contain Ethernet or IP header fields.

SNMP Extended Agent Demo

The demo can be run on a PC running Windows or Linux platform.

Demo Setup

The extended agent demonstration can be run with the following components.

1. SNMP network manager – MIB browser, provided by customer. The demo was tested with ManageEngine MIB browser, but other MIB browsers can be used.
2. xPico 200 firmware image built with the optional SNMP module. For details on building the firmware, see the latest xPico 200 SDK Guide or XPort EDGE SDK Guide.
3. SNMP agent simulator, provided by customer. The demo was tested with a licensed version of Mimic® SNMP Simulator by Gambit Communications, which is required to collect the custom MIB variables. An open source Linux SNMP agent can be used as simulator to demonstrate the standard limited MIBs.
4. Host script – Python script to receive and process the data according to the connection type and handle type input. For details of the script, see *Host Script*.
5. MIB files – provided by the customer and loaded onto the SNMP agent simulator and the SNMP Network Manager MIB browser.

Figure 3 shows the extended agent demo setup.

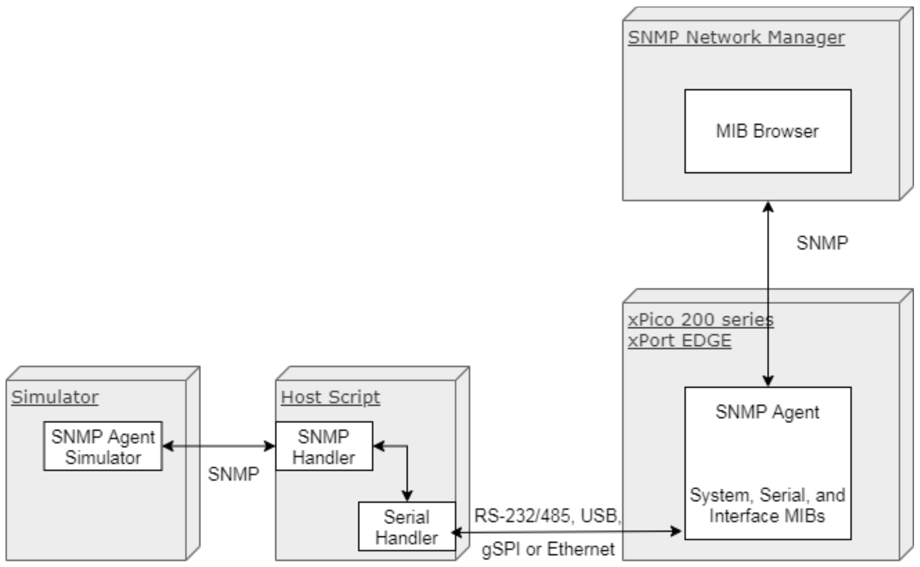


Figure 3. Extended Agent demo setup

Run the Demo

The demo can be run on a PC running Windows or Linux platform. Before you follow the procedure below, install the required packages for Python 3, pyserial, SNMP agent and MIB browser on the PC.

In this demo, the xPico 200 series device is connected via the serial interface to the SNMP agent simulator on a host PC. Line 1 protocol is configured for Tunnel, although Mux could be used or the connection type could be ethernet. For further assistance, please contact Lantronix Technical Support.

To run the demo:

1. Load the SNMP enabled SDK rom file into the xPico device.
2. Configure SNMP on the required interface and enable the extended agent by configuring the Extended Agent Port (host port) number to communicate with the tunnel through the loopback interface.
    a. In Web Manager, go to **SNMP** and configure the SNMP settings. The SNMP agent state must be Enabled. Enter the Extended Agent IP address (loopback address) and port number to enable the extended agent. Enter other SNMP settings as necessary to configure SNMP.
    b. In Web Manager, go to **Line** and configure Line 1 protocol to Tunnel.
    c. In Web Manager, go to **Tunnel** and configure Tunnel 1 Accept connection. Set the protocol to TCP and the local port to the Host Port number.
    d. ***Note:*** *You can observe the SNMP extended agent connected to the loopback IP address by viewing the tlog. (Type <IPaddress>/tlog in the browser and view* `SNMP Extd. Agent Connected to <loopback ip address:host-port>`*).*

3. Connect the xPico 200 serial line to the PC.
4. Create a simulated device in the SNMP Agent simulator and load the MIB files that are provided with the demo zip file.
5. In the host script, configure the SNMP agent simulator IP address and port. See *Host Script Configuration* below for details of the configurable parameters.
6. Run the Python host script and include the following arguments: script file name, connectionType, and handleType. Example usage for command line follows:

```
python .\snmpHostScript_1.2.py serial host
```

6. Open the MIB browser.
    a. Configure the SNMP extended agent IP address.
    b. Load the MIBs provided in the sample demo files.
    c. Use the SNMP request commands to get the OID values. The following SNMP commands are supported: Get, Set, Get next, Walk, and V2Traps.
    d. View the response in the MIB browser.

Host Script Configuration

The host script arguments and configuration settings are described in the following table.

Field	Description
HANDLE_TYPE	The handle type argument determines how the host script will process the data. Options are *auto, Lantronix* or *Host*. The default is *auto*. • *lantronix* - Lantronix SNMP agent behavior is expected. If the handle type is "lantronix", the script echoes the serial data by inserting "00" after "SNMP". • *host* - Extended agent behavior is expected. If the handle type is "host", then it converts the serial data to SNMP request and sends it to the SNMP agent simulator. At the same time, it converts the SNMP response received from the SNMP agent simulator to serial SNMP packet, then inserts "01" after "SNMP". • *auto* – the script runs as *host* type. If the handle type is "auto", it will process the data as "host" handle type and if the SNMP agent simulator generates any failure, it automatically switches to "lantronix" handle type.
CONNECTION_TYPE	The connection type argument describes the connection between the xPico device and the host processor. Options are *tcp* or *serial*. • *tcp* – for connections over eth0 • *serial* – for serial connections, including RS-232/RS-485, USB, or gSPI.
SERIAL_PORT	Configure the serial port for the PC platform in use and comment out the unused platform. Default values are Windows = `"COM1"` and Linux = `"/dev/ttyS0"`.
SERIAL_BAUD	The baud rate setting for the serial connection. This must match the baud rate of the xPico 200 device. Default rate is 115200.

Field	Description
UDP_IP	IP address of the SNMP Agent simulator.
UDP_PORT	Port number of the SNMP Agent simulator.

Host Script

Version 1.2

The host script is shown below.

```
#!/usr/bin/python3
import socket
import sys
import serial
import os
import errno
import time, subprocess

SERIAL_PORT = "/dev/ttyS0" #For Linux
#SERIAL_PORT = "COM4"  #For Windows

HANDLE_TYPE = 'auto'

SERIAL_BAUD = 115200
SNMP_SIM_IP = "10.4.152.1"
SNMP_SIM_PORT = 161

if len(sys.argv) < 3:
    print("Usage: " + os.path.basename(sys.argv[0]) + " <Connection Type
[serial/tcp]> <Handle Type [lantronix/host/auto]>")
    sys.exit()

else:
    if '-h' == sys.argv[1]:
        print("Usage: " + os.path.basename(sys.argv[0]) + " <Connection Type
[serial/tcp]> <Handle Type [lantronix/host/auto]>")
        sys.exit()
    connectionType = sys.argv[1]
    if 'serial' != connectionType and 'tcp' != connectionType:
        print("Invalid Connection Type")
        print("Usage: " + os.path.basename(sys.argv[0]) + " <Connection Type
[serial/tcp]> <Handle Type [lantronix/host/auto]>")
        sys.exit()
    handleType = sys.argv[2]
    if 'lantronix' != handleType and 'host' != handleType and 'auto' !=
handleType:
        print("Invalid Handle Type")
        print("Usage: " + os.path.basename(sys.argv[0]) + " <Connection Type
[serial/tcp]> <Handle Type [lantronix/host/auto]>")

serialSnmpData = 0
ipAddress = 0
portNumber = 0
iteration = 1
```

```
readData = 0
data = 0

if (handleType == 'host' or handleType == 'auto'):
# Create a UDP socket for SNMP
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server_address = (SNMP_SIM_IP, SNMP_SIM_PORT)
    sock.setblocking(0)
    #fcntl.fcntl(sock, fcntl.F_SETFL, os.O_NONBLOCK)

if connectionType == 'serial':
#init serial port
    serialPort = serial.Serial(port=SERIAL_PORT,baudrate=SERIAL_BAUD,
timeout=.0125)
    serialPort.WriteBufferSize = 256

elif connectionType == 'tcp':
# Create a TCP/IP socket
    serverSock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# Bind the socket to the port
    serverAddress = ('10.4.152.1', 10001)
    serverSock.bind(serverAddress)
    serverSock.listen(2)
    print("Waiting for client to be connect")
    clientConn, clientAddress = serverSock.accept()
    print("client connected: ", clientAddress)

while True:
    try:
        if connectionType == 'serial':
            if (serialPort.inWaiting()>20):
                readData=serialPort.read(255)
        elif connectionType == 'tcp':
            try:
                readData = clientConn.recv(256)
            except IOError as e:
                if e.errno == errno.EWOULDBLOCK:
                    pass

        if readData:
            print("\nIteration-", iteration)
            iteration += 1
            print("Received Data: ", readData)
            sofOffset = readData.find(b'SNMP')
            snmpData = readData[sofOffset:len(readData)]
            if b'SNMP' == snmpData[0:4]:
                length = int(snmpData[10]) * 100 + int(snmpData[11])
                if (length == (len(snmpData) - 12)):
                    ipAddress = snmpData[4:8]
                    portNumber = snmpData[8:10]
                    serialSnmpData = snmpData[4:(len(snmpData))]
                    if handleType == 'lantronix':
                        sendData = b'SNMP' + b"\0" + serialSnmpData
                        print("\nData to be send: ", sendData)
                        if connectionType == 'serial':
                            serialPort.write(sendData)
                            readData = 0
                            continue
                        elif connectionType == 'tcp':
                            clientConn.sendall(sendData)
                            readData = 0
                            continue
                    elif (handleType == 'host' or handleType == 'auto'):
                        sock.sendto(snmpData[12:(len(snmpData))], server_address)
                        print("data sent to SNMP agent")
                else:
                    print("Error: Length field mismatch with the received data
length")
            else:
                print("Error:SOF is not 'SNMP'")

        if (handleType == 'host' or handleType == 'auto'):
            try:
                time.sleep(1)
                data, ip = sock.recvfrom(512)
                if data:
                    print("\nSNMP Data: ", data)
                    errorLength = 17 + int(data[6])
                    if (handleType == 'auto' and int(data[errorLength]) == 1):
                        sendData = b'SNMP' + b"\0" + serialSnmpData
                    else:
                        lengthMSB = round(len(data) / 256)
                        lengthLSB = len(data) - lengthMSB * 256
                        lengthData = [lengthMSB, lengthLSB]
                        sendData = b'SNMP' + b"\1" + ipAddress + portNumber +
bytes(lengthData) + data

                    print("\nData to be send: ", sendData)
                    if connectionType == 'serial':
                        serialPort.write(sendData)
                        serialPort.flushInput()
                        readData = 0
                    elif connectionType == 'tcp':
                        clientConn.sendall(sendData)
                        readData = 0
            except IOError as e:
                    if e.errno == errno.EWOULDBLOCK:
                        pass

    except KeyboardInterrupt:
        print("Keyboard Interrupt detected to stop the script")
        if clientConn:
            print("Closing client connection")
            clientConn.close()
        if serverSock:
            print("Closing Socket")
            serverSock.close()
        sys.exit()
```