InHand OpenDevice Platform User Manual

Introduction

InHand OpenDevice is a basic platform based on InRouter900 industrial 4G router. Using Python, it provides a programmable environment for users to develop their applications easily. InHand OpenDevice platform provides the following services:

System Management

The system management service is responsible for controlling and monitoring of user applications. Customized Application can be launched and terminated by system management via webserver. Besides, system management service can also monitor the running status of applications via logs. Based on different demands, users can set different log levels and define the event processing of relevant log level, for example, alarm will be triggered if error logs are detected.
Based on InHand DeviceTouch, users not only can monitor the running status of applications in the router which is installed at industrial field, but also can carry out remote deployment, debugging and application upgrade.

OpenDevice Platform API

Using API, customized applications can configure parameter of router physical interface, such as serial port and I/O, it can also inquire device information and interface status, such as router firmware version, cellular interface status and so on. Besides, OpenDevice API provide asynchronous program interface, application of users can subscribe status message of physical interface and prepare corresponding processing methods according to demands. Take cellular module as example, when cellular module status of router transfers to offline from online, application of users will receive relevant notification, and corresponding handle should be executed.

Secure Access to Device Networks Cloud

InHand OpenDevice platform furnishes customized application with secure access to

InHand Device Networks Cloud. Customized application can upload collected data to the Device Networks Cloud, After that, those data can be carried out further analysis by using the service of the Device Networks Cloud. Besides, by using Device Networks Cloud, customized applications in router which is deployed at different industrial field can be controlled and monitored.

Local Data Caching

InHand OpenDevice platform provides data caching service which encapsulates Sqlite3. Users can select NAND Flash of router or TF card as extended storage.

Message Queue

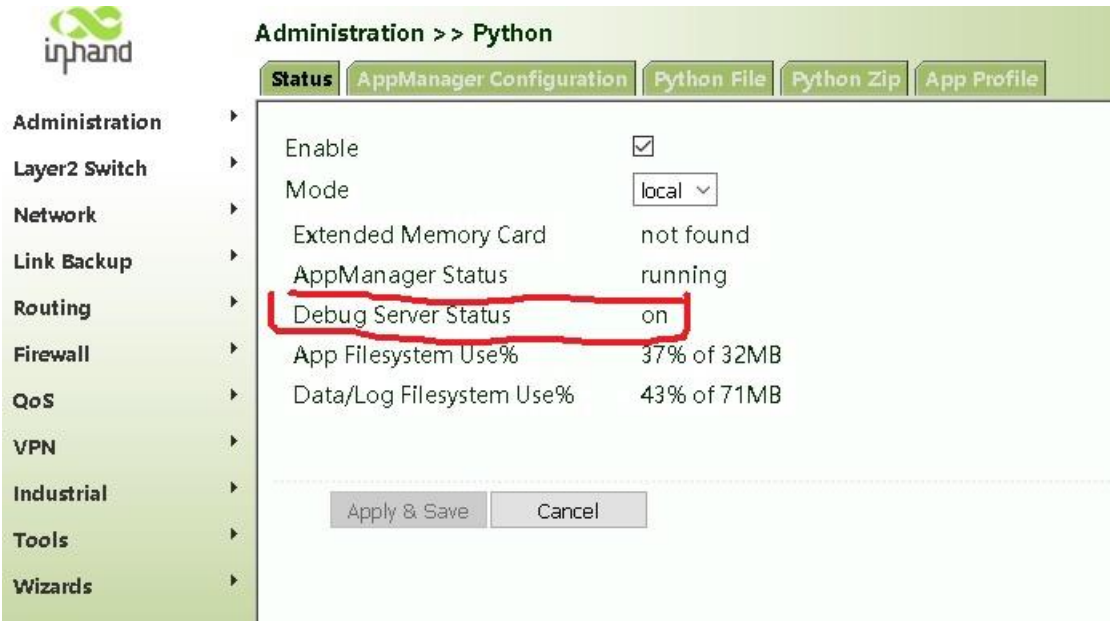
InHand OpenDevice platform employs Nanomsg message queue, which is achieved by C language, with BSD-Socket-like API, less memory used and CPU occupancy, thread-safe and so on. InHand OpenDevice platform provides various communication model which is encapsulated in Python class based on Nanomsg message queue, such as publish/subscribe model, request/response model and so on. Customized applications can instantiate these python classes, inherit or overload them.

Customized Application Debugging

InHand OpenDevice platform can provide debugging tools kit for customized applications. For example, based on Windows, users can use PuTTY to verify applications after login the router via WinSCP. The detailed operation steps are as follows:

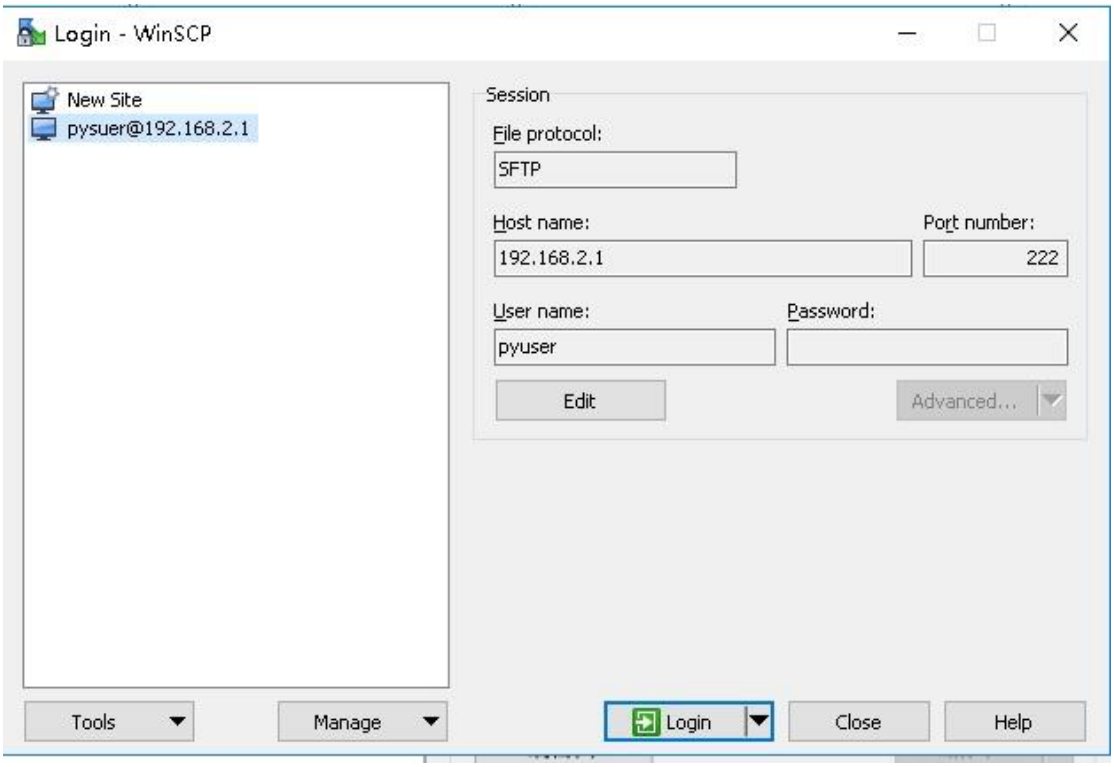
1. Enable debug server

Under CLI configuration view, execute “python start local” and “python debug on”. After correctly configured, status page is shown below after debug server is correctly enabled:

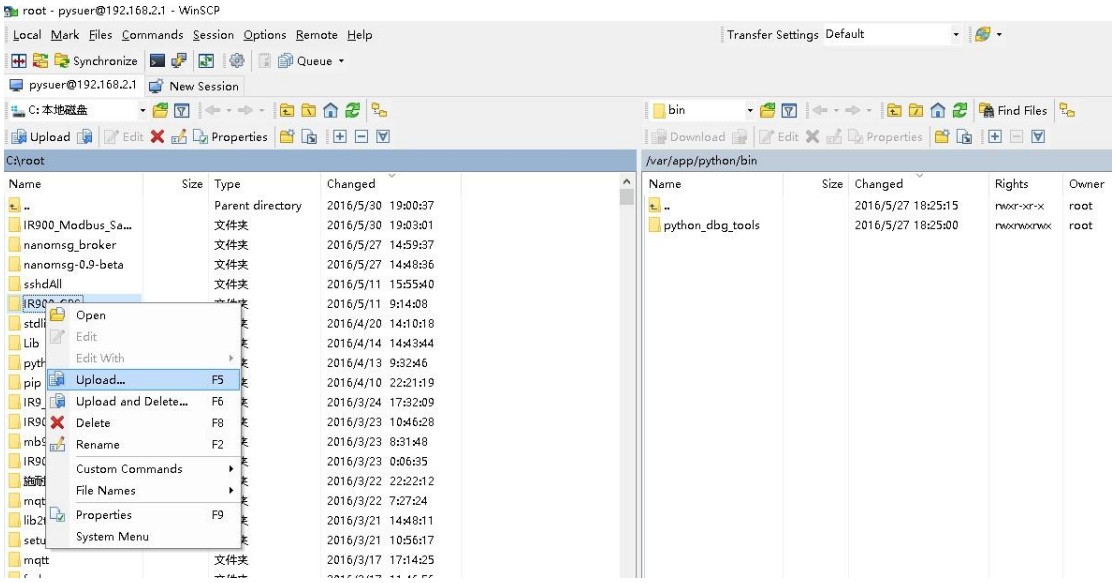


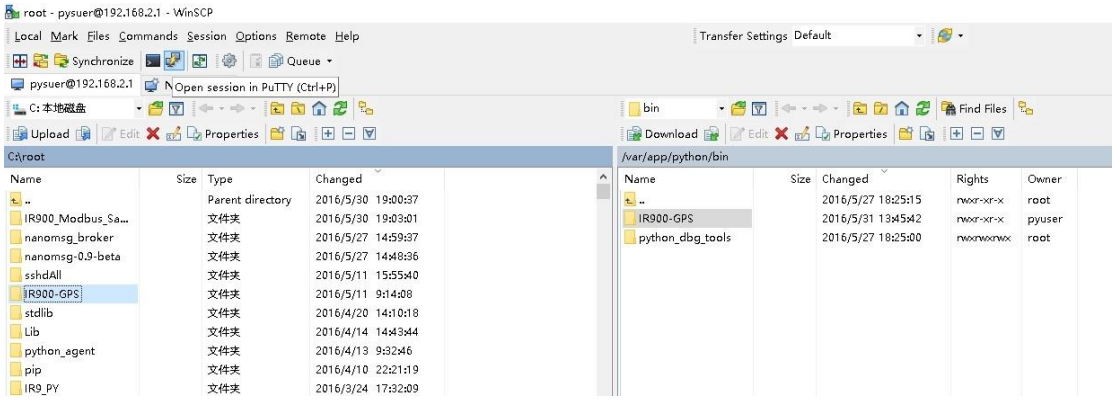
2.Connect to debug server

Under Windows development environment, it is recommended to connect WinSCP to debugging server and synchronize Python application files and carry out application debugging by using PuTTY. After open WinSCP, select file protocol SFTP and configure a host name being LAN address of router. In the example, address of the router is 192.168.2.1, with a configured port number of 222, user name of pyuser and password of 123456 as shown in the figure below:

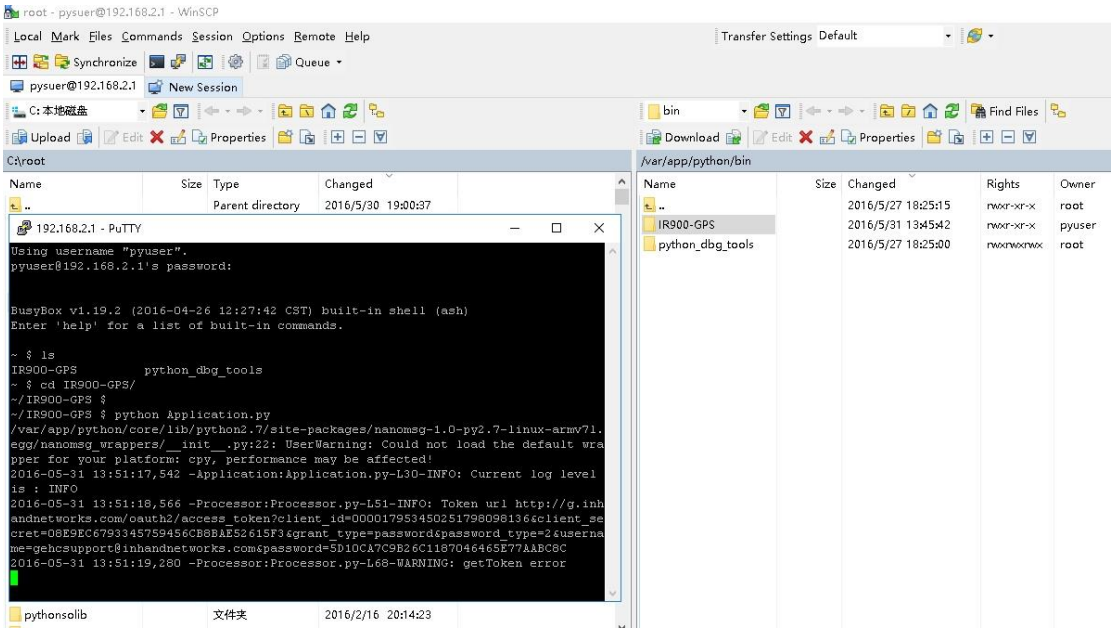


After login is succeeded, select the Python application that will be uploaded, and press button F5 or click right button to upload it. For example, upload IR900GPS directory, after file synchronization is succeeded, files uploaded on the equipment can be seen on right interface of WinSCP. Click *enable PuTTY* button in WinSCP to enable PuTTY. If PuTTY is not installed in default path, please specify PuTTY.exe path in WinSCP settings. Besides, WindSCP can also provide command line, which can be directly used by users. As shown in figure below:



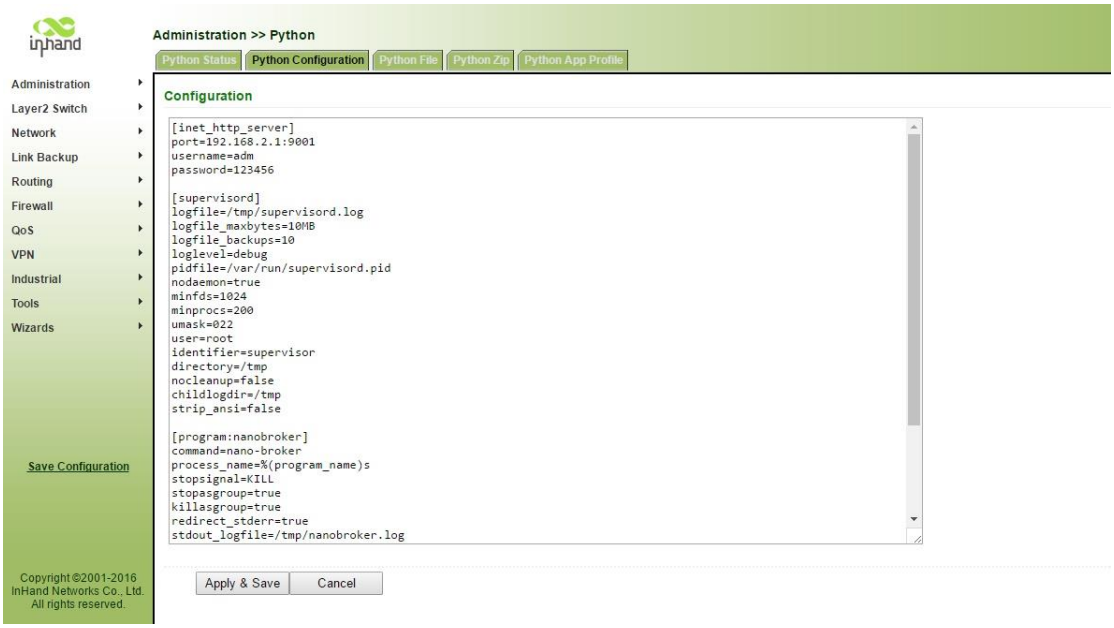


After IR900GPS directories synchronized correctly , launch PuTTY and input password **123456** and **ls** command. Python application can be debugged by executing *cd IR900GPS* to enter the directories and inputting *python Application.py*.



3. AppManager configuration

After customized application running correctly, users should add this application auto-start configuration to AppManager. New application auto-start configuration as shown below:



As shown in the figure above, at least two configurations are required. Application name will be the first, such as *[program:IR900GPS]*, the application name should be unique; and then, the next is application path, such as *command=python /var/app/python/bin/IR900GPS/Application.py*.

Customized Application Upload

InHand OpenDevice platform supports to add customized application in py singlefile mode and multiply file zip package. Users can upload customized application in two ways: via Web service or CLI.

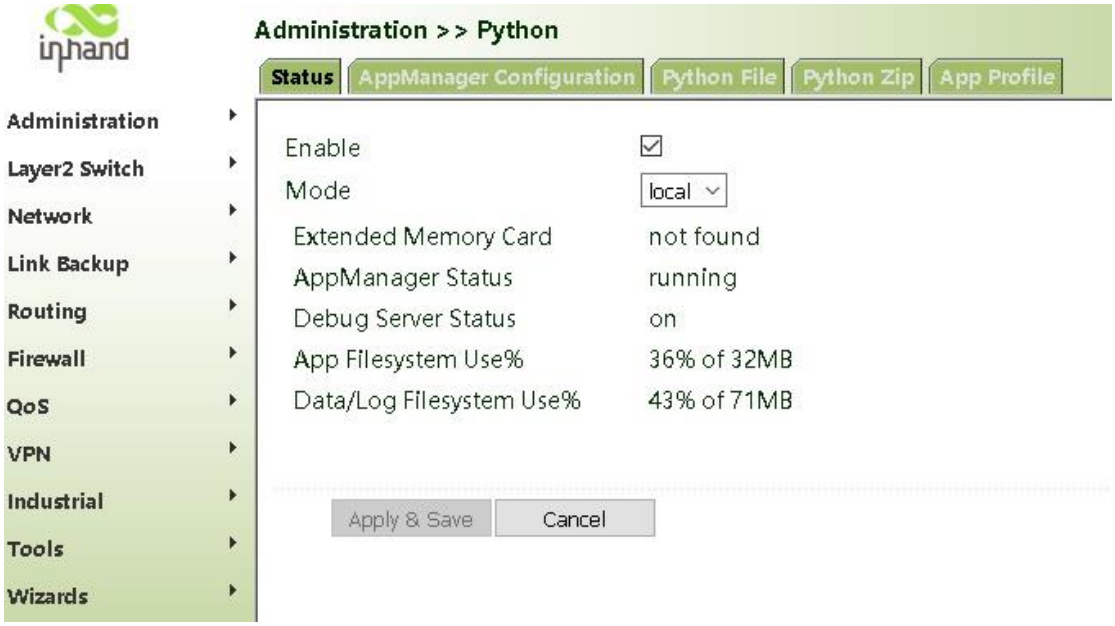
Precautions for Python uploading files

- Single Python file shall not be more than 625KB
- Python file zip package is zipped in zip format
- Python folder name should be consistent with that of generated zip package; for example, name of generated package of pyserial directory should be pyserial.zip
- Before upload applications, it should be ensured that InHand OpenDevice platform is correctly installed.

Customized Application Upload via Web

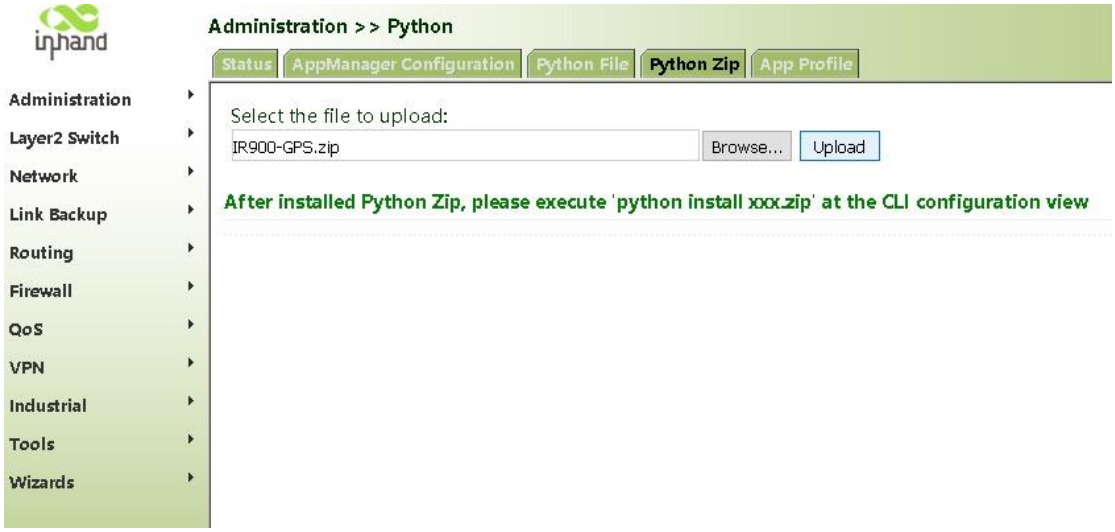
1. Enable InHand equipment platform

Login web server of router and click Administration>Python>Status to enable OpenDevice service.



2. Upload Python Zip package

Customized application which is packed into a Zip package can be uploaded by clicking label Python Zip; if there is only one Py file of user item needed to be uploaded, click Python File. By taking Python Zip package with name of IR900GPS.zip as example, see the figure below:

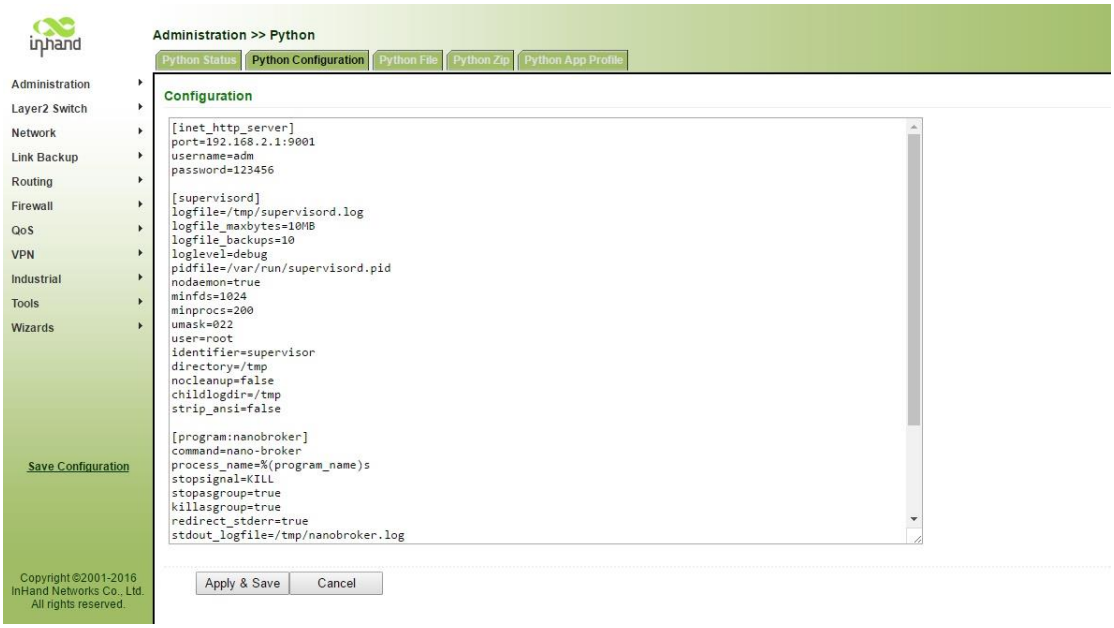


Select label *Python File* and click Browse to select the file needed to be uploaded, and then click *Upload* to upload py file.

After file uploading is succeeded, users should execute python install IR900_GPS. Zip under configuration view of CLI.

3. AppManager configuration

After Python Zip package is successfully installed, users should add new application auto-start configurations. Click label *AppManager configuration* to add new application start configurations. As shown in the figure below:



As shown in the figure above, at least two configurations are required for application selfstarting adding. Application name will be the first, such as *[program:IR900GPS]*, the application name should be unique; and then, the next is application path, such as *command=python/var/app/python/bin/IR900_GPS/Application.py*.

Customized Application Upload via CLI

1.Upload py file or zip files

For uploading customized application, tftp/ftp server on the PC is required, and py file requiring to be imported need to be placed in uploading directories of server. When customized application is composed of many py files, directories containing py files should be zipped as a zip file and placed on uploading directories and uploaded via “copy” command in the configuration view of CLI. For example, import zmqpubsub.py files. As shown in the figure below:

```
22:42:29 Router# configure terminal
22:42:44 Router(config)# copy tftp: python zmqpubsub.py
Address or name of remote host []:192.168.2.222
Source filename []:zmqpubsub.py
copy python py file to /var/app/python/bin/zmqpubsub.py

22:43:04 Router(config)#
22:43:09 Router(config)# show python script
zmqpubsub.py
22:43:10 Router(config)#
22:43:11 Router(config)#
22:43:11 Router(config)#
```

Uploaded files can be checked via show python script command, which indicates that

zmqpubsub.py has been successfully uploaded.

2.Install Customized Application

Customized application which is packed in zip file should be executed python install under configuration view of CLI. For example, after successfully uploading IR900_Modbus.zip, “python install IR900_Modbus.zip” should be executed as shown in the figure below:

```
14:19:24 Router(config)#
14:19:27 Router(config)# copy tftp: python IR900_Modbus.zip
Address or name of remote host [1.1.1.1]:192.168.2.222
Source filename [asdasd]:IR900_Modbus.zip
copy python zip file to /var/python/IR900_Modbus.zip
cli exec cmd unzip -oq /var/python/IR900_Modbus.zip

14:20:12 Router(config)#
14:20:13 Router(config)#
14:20:13 Router(config)#
14:20:14 Router(config)#
14:20:43 Router(config)# python install IR900_Modbus.zip
14:20:47 Router(config)#
```

3.Customized Application Edit

If new customized application is required to be created, be modified or uploaded, users can use python edit command for creation or edition. For example, edit zmqpubsub.py as shown below:

```
22:44:41 Router(config)# python edit zmqpubsub.py
        logging.info("zmq sub handle running...")

        context = zmq.Context()
        socket = context.socket(zmq.SUB)

        socket.connect(connect_dst)
        logging.info("zmq connect to %s" %connect_dst)

        if not bind_topic:
                socket.setsockopt(zmq.SUBSCRIBE, '')
                logging.info("Receiving all  messages")
        else:
                logging.info("Receiving on topics %s" %bind_topic)
                for topic in bind_topic:
                        socket.setsockopt(zmq.SUBSCIRBE, topic)
                        logging.info("Receiving on topic %s" %topic)

        while True:
                msg = socket.recv()
                logging.info("recv msg %s" %msg)

if __name__=="__main__":
        main()
```

For verifying customized application “zmqpubsub.py” , *python run zmqpubsub.py* can be executed in command line.

4.AppManager Configuration

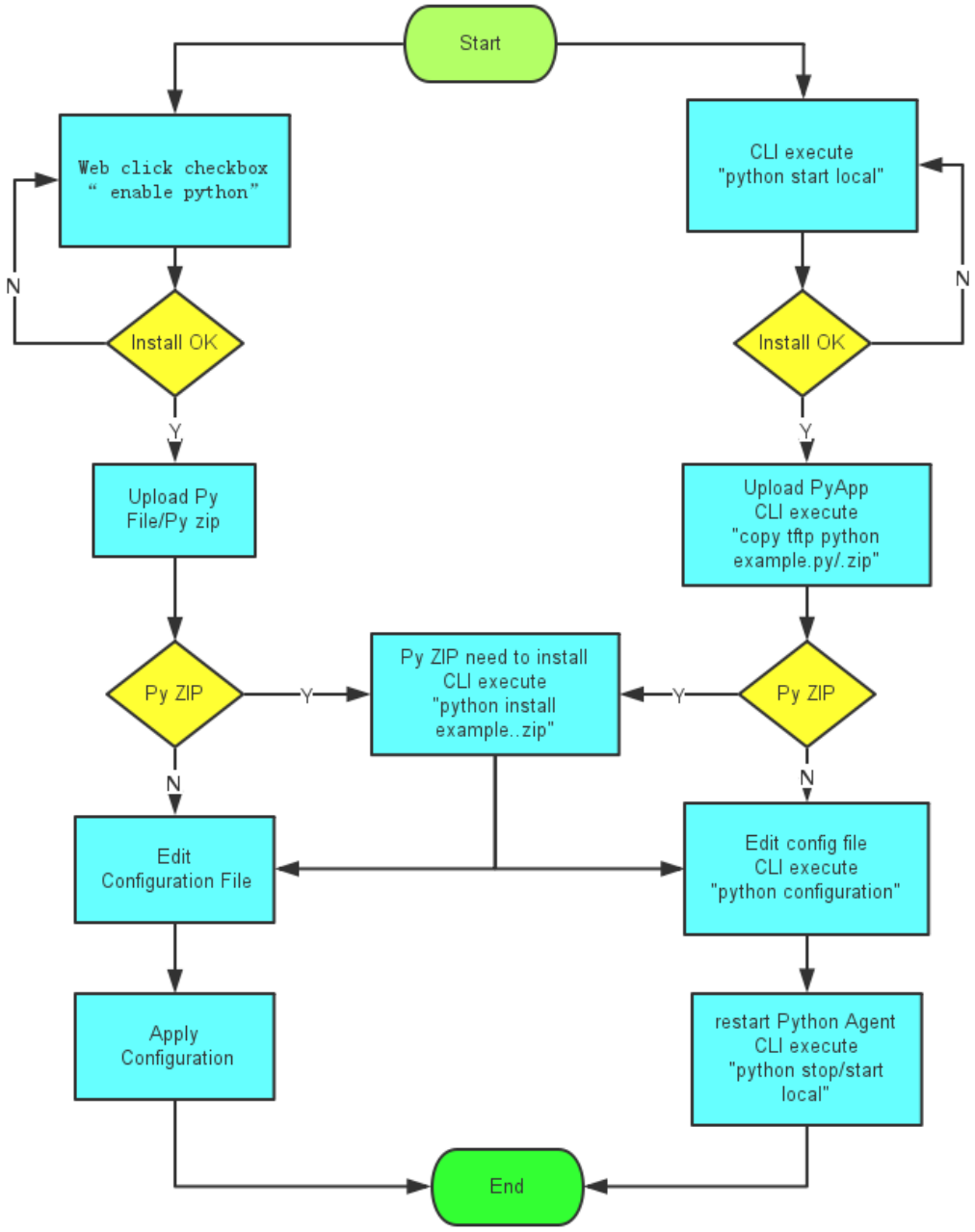
Under configuration view of CLI, execute “python configuration” to edit configuration files. For example, start zmqpusub.py. As shown below:

```
[program:zmqpubsub]
command=/var/app/python/bin/zmqpubsub.py tcp://127.0.0.1:5566
process_name=%(program_name)s
stopsignal=KILL
stopasgroup=true
killasgroup=true
redirect_stderr=true
stdout_logfile=/tmp/zmqpubsub.log
stdout_logfile_maxbytes=1MB
stdout_logfile_backups=1
stdout_capture_maxbytes=1MB
stdout_events_enabled=false
stderr_logfile=/tmp/zmqpubsub_err.log
stderr_logfile_maxbytes=1MB
stderr_logfile_backups=1
stderr_capture_maxbytes=1MB
stderr_events_enabled=false
environment=A="1",B="2"
serverurl=AUTO
I /var/app/python/core/cfg/supervisord.conf 62/62 100%
```

After modified AppManager configuration, execute “python restart” under configuration view to apply new startup configuration.

Customized Application Upload Process

Flowchart for upload customized application is shown below:



InHand OpenDevice Platform CLI reference

Command	Description	Comment
python start local	Start InHand OpenDevice platform	If OpenDevice is not installed, try to find installation file from local storage and external memory card. If the process is retried and failed for 3 times, it

		indicates that the installation is failed.
python stop	shutdown InHand OpenDevice platform	
python restart	restart InHand OpenDevice platform	
python reset	cleanup InHand OpenDevice platform	remove all customized applications installed on router, as well as collected data and running logs which is generate by customized applications
python debug on	start debug server	
python debug off	terminate debug server	
python configuration	Edit AppManager Configuration	
copy tftp python filename	Upload customized application	Upload customized application by using tftp protocol. format of upload file such as py,pyc and zip can be suppoted
python install zipfile	install customized application in zip package	
python remove filename	Delete Python files and zip package	Delete the imported Python files and zip packages in router, with filename being that of py/pyc files of Python or zip package of Python files.
python setup packagename	Setup python package which is upload from local	
python pip install packagename	Setup python package using pip	
python pip uninstall packagename	Uninstall python package using pip	
python pip list	list all installed python package information	
python pip show	show installed python package	

packagename	information which is named packagename	
python run [application]	launch python command line shell	if application is configured, try to execute correspondent application
show python status	show OpenDevice status	
show python script	show all uploaded python file	
show python zip	show all uploaded python zip package	
show python configuration	show AppManager configuration	

InHand Networks

InHand Networks provides reliable, secure and intelligent M2M solution for electric power, industrial automation, commercial and medical devices. InHand Networks is recognized by world class customers and partners and expanding with intensive investments in research and development.

InHand Networks has become leader in industrial grade network technology by providing industrial cellular routers, industrial Ethernet switches, wireless sensor network devices and cloud based M2M platforms.

Connecting devices, enabling services.

InHand Networks

3900 Jermantown Rd., Suite 150
Fairfax, VA 22030
USA
T: +1-703-348-2988
F:+1-703-348-2988
www.inhandnetworks.com
Inquiry: info@inhandnetworks.com
Technical Support: support@inhandnetworks.com