



InHand Networks IG902

API

Reference MANUAL

Version: V1.0 Date: 2019.03

InHand Networks
Global Leader in Industrial IoT
www.inhandnetworks.com

Glossary

Abbreviation	Full Name
DN	Device Networks
IDE	Intergrated Developement Environment
SDK	Software Development Kit

RouterInfo Module

The RouterInfo module can be used to obtain router information such as dialer module information, GPS information, router firmware version information, and so on. This module can also be used to configure gateway devices through CLI commands. The sample code of the RouterInfo module API is as follows:

```
# !/usr/bin/env python
# -*- coding: utf-8 -*-
# Based on nanomsg and libevent
'''
App example
@author: InHand
'''

import os
import time
import json
import logging
import libevent
from RouterInfo import RouterInfo
from AppTemplate_libevent import AppTemplate
from MessageChannel import MessagePush, INOS_PUSH_ADDR

INPY_APP_PATH = '/var/app/'

class AppExample(AppTemplate):
    def __init__(self, vendor_name, app_name, version='1.0.0'):
        AppTemplate.__init__(self, vendor_name, app_name, version,)
        self.ri = RouterInfo(self.base)
        self.timer = libevent.Timer(self.base, self.timer_handler, userdata=None)
        self.app_name = app_name
        self.logger._logger.setLevel(logging.DEBUG)

    def timer_handler(self, evt, userdata):
        self.logger.info(self.ri.get_cellular_info()) # Get device cellular information
        self.logger.info("*****50)
        self.logger.info(self.ri.get_gps_info()) # GPS
        self.logger.info("*****50)
        self.logger.info(RouterInfo.get_firmware_info())
```

```

self.logger.info(""*50)
self.logger.info(RouterInfo.get_local_time())
self.logger.info("time: %s" % time.time())
self.timer.add(3)

def init(self):
    self.running_conf = None
    self.timer.add(1)

if __name__ == '__main__':
    app = AppExample('InHand', '')
    app.init()
    app.run()

```

The RouterInfo module API is described in detail below:

class RouterInfo

Instantialize a class object ri = RouterInfo(base)

Parameter

base: libevent.Base() instance object

Class Python member methods are as follows:

get_cellular_info()

Parameter

Null

Interface description

Get dial module information

Return value

```

{
  "mcc": 460,
  "rssi": 26,
  "operator": "China Unicom",
  "iccid": "89860111811011851448",
  "mnc": 1,
  "imsi": "460011050002097"
}

```

get_gps_info()

Parameter

Null

Interface description

Get GPS information

Return value

```
{
  "gps_time": "2017-06-21T16:17:06+08:00",
  "latitude": 39.925755000000006,
  "speed": 0.407,
  "longitude": 116.19064833333334
}
```

The unit of speed is Knots (1knot = 1.85km/h)

get_dataflow()

Parameter

Null

Interface description

Get current traffic information

Return value

```
{34237, 653456}
```

The first element in the array is the number of bytes received, and the second element is the number of bytes sent.

get_local_time()

Parameter

Null

Interface description

Get the current time of the device

Return value

```
'2019-03-04T15:21:27+08:00'
```

get_firmware_info()

Parameter

Null

Interface description

Obtain gateway firmware version information

Return value

```
{
  "lang": "Chinese",
  "hostname": "EdgeGateway",
  "timezone": "UTC-8",
  "model_name": "902B",
  "oem_name": "inhand",
  "bootloader": "2017.01.r10319",
  "serial_number": "IG9021900434193",
  "product_number": "EN00",
  "description": "www.inhandnetworks.com",
  "sw_version": "1.0.0.r10403",
  "encrypt_passwd": "no"
}
```

get_connect_device()

Parameter

Null

Interface description

Get device information connected to the gateway

Return Value

```
[
  {"ip": "192.168.1.111", "mac": "b8:70:f4:da:cd:1b", "hostname": "", "dev": "fastethernet 0/1"},
  {"ip": "192.168.30.30", "mac": "00:26:55:3a:68:b1", "hostname": "", "dev": "bridge 1"}
]
```

get_dhcp_device()

Parameter

Null

Interface description

Obtain the device information managed by the gateway DHCP

Return value

```
[  
  ["br1", "44:37:e6:59:8b:82", "192.168.30.196", "ES07734857", "0 day, 23:21:53"]  
]
```

get_app_info()

Parameter

Null

Interface description

Get device installation app information

Return value

```
[  
  {"version": "0.0.1", "name": "GatewayInfoSample", "sdkVer": "1.3.2"}  
]
```

set_openvpn_state(conf, state):

Parameter

Conf refers to the OpenVPN configuration file path, it need to fill in the absolute path State Sets the state of OpenVPN. If you create an OpenVPN tunnel, set state='on', otherwise set state='off'

Interface description

Create/close an OpenVPN tunnel

Return value

Null

do_reboot():

Parameter

Null

Interface description

Reboot the device

Return value

Null

get_wlan_scan():

Parameteer

Null

Interface description

Get wlan scan information

Return value

```
[
  {'signal': '-46', 'frequency': '2412', 'sec': '[WPA-PSK/AES] [WPA2-PSK/AES]', 'ssid':
  'Inhand', 'bssid': '84:a9:c4:5f:6c:61'},
]
```

get_cert_status():

Parameter

Null

Interface description

Get status information of the current certificate request

Return value

```
{
    "status": "resume",
    "describe": "Resume Certificate",
    "timestamp": "2017-06-21T16:17:06+08:00"
}
```

get_fw_nat():

Parameter

Null

Interface description

Get the current firewall nat rule

Return value

```
['ip nat static 1.1.1.1 2.2.2.2', 'ip nat static 3.3.3.3 interface fastethernet 0/1']
```

get_connection_type():

Parameter

Null

Interface description

Get current network connection status

Return value

1, 2, 3 or -1

1: cellular 2:ethernet 3:wifi -1:error

set_network_track(index, addr, frequency, timeout):

Parameter

index Index of sla addr Destination address frequency Detection frequency timeout Detection timeout

Interface description

Set up link detection

Return value

True or False

True: success False:fail

get_network_track():

Parameter

Null

Interface description

Get link detection status

Return value

```
[{'tid': 1, 'stat': 0}]
```

tid : track id; stat: {0:inactive ; 1:active; -1:erro}

InDB Module

The InDB module provides users with a class encapsulated API for data caching operations. The sample code is as follows:


```
import InDB

if __name__ == '__main__':
    dbPath = '/var/app/data/test.db'
    db = InDB.InDB(dbType=InDB.DB_TYP_SHARED, dbRole=InDB.DB_ROL_OWNER)
    db.open(dbPath)
    # Initialization table
    data_tb = db.get_table('data', tblType=InDB.TBL_TYP_KV)
    # Deposit data
    data_tb.put("1", "hello_ChengDu")
    # Take out data
    print data_tb.get("1")
```

The detailed description is as follows:

class InDB

Initializing parameters

```
dbType:
    - DB_TYP_PRIVATE:
        this DB is only accessed by the own process itself. Note that any persistent DB will
        be accessed by the system manage process. See the dbPersistent arg below.
    - DB_TYP_SHARED:
        this DB is used to shard data with multiplie processes.

dbRole:
    - DB_ROL_OWNER:
        the present process own this database. It's the owner's responsibility to create
        database and tables before the others using them. For a private DB, the one who open it
        is forced to be owner.
    - DB_ROL_WRITER:
        the one who can write or read this database.
    - DB_ROL_READER:
        the one who can only read this database.
```

Operation example

```
db = InDB.InDB(dbType=InDB.DB_TYP_SHARED, dbRole=InDB.DB_ROL_OWNER)
```

Open database

open(self, dbPath)

Parameter

dbPath: Database storage path

Return value

Null

Operation example

```
db.open(dbPath)
```

Determine if the database is open

is_opened(self)

Parameter

Null

Return value

True or False

True: open database successfully False: open database unsuccessfully

Operation example

```
db.is_opened()
```

Close database

close(self)

Parameter

Null

Return value

Null

Operation example

```
db.close()
```

Get database table

get_table(self, name, tblType)

Parameter

name: The name of data table

tblType:

- TBL_TYP_KV: Key-value pair type data table
- TBL_TYP_TS: Time series type table

Return value

Table instance

Operation example

```
data_tb = db.get_table('data', tblType=InDB.TBL_TYP_KV)
```

Insert data

put(self, key, val)

Parameter

key: Keys that need to store data Val: Data need to store, json

Return value

None

Operation example

```
key = 'power'  
val={'addr':40001, 'timestamp':12341234, 'value':321}  
data_tb.put(key, val)
```

Read data

get(self, key)

Parameter

key: Get the key to store data

Return value

The stored data corresponding to the keyword, no data returns None

Operation example

```
key = 'power'  
val = db.get(key)
```

Pop data

pop(self, key)

Parameter

key: Keys that need to store data

Return value

The stored data corresponding to the keyword, no data returns None

Operation example

```
key = 'power'  
val = db.pop(key)
```

Delete data

delete(self, key)

Parameter

key: Key to delete data

Return value

Null

Operation example

```
key = 'power'  
db.delete(key)
```

Get all the keys in the database

keys(self)

Parameter

Null

Return value

Store data key value list

Operation example

```
db.keys()
```